SIEMENS

SIMATIC PCS 7 OSx

Interface to S7 Controllers

User Manual

Order Number: 6ES7 6550XX058BC8
Manual Assembly Number: 2811155–0001
Original Edition

⚠ DANGER

DANGER indicates an imminently hazardous situation that, if not avoided, will result in death or serious injury.

DANGER is limited to the most extreme situations.

⚠ WARNING

WARNING indicates a potentially hazardous situation that, if not avoided, could result in death or serious injury, and/or property damage.

⚠ CAUTION

CAUTION used with a safety alert symbol indicates a potentially hazardous situation that, if not avoided, could result in minor or moderate injury.

CAUTION

CAUTION used without the safety alert symbol indicates a potentially hazardous situation that, if not avoided, could result in property damage.

NOTICE

NOTICE indicates a potential situation that, if not avoided, could result in an undesirable result or state.

MANUAL PUBLICATION HISTORY

SIMATIC PCS 7 OSx Interface to S7 Controllers
Order Manual Number: 6ES7 6550XX058BC8
Refer to this history in all correspondence and/or discussion about this manual.

Event	**Date**	**Description**
Original Issue	7/02	Original Issue (2811155–0001)

LIST OF EFFECTIVE PAGES

Pages	Description	Pages	Description
Cover/Copyright	Original		
History/Effective Pages	Original		
iii — xv	Original		
1-1 — 1-23	Original		
2-1 — 2-50	Original		
3-1 — 3-35	Original		
4-1 — 4-42	Original		
A-1 — A-2	Original		
B-1 — B-6	Original		
C-1 — C-14	Original		
D–1 — D-12	Original		
Index-1 — Index-7	Original		
Registration	Original		

Trademarks

Contents

List of Figures

List of Tables

Preface

New Features of PCS 7 OSx

SIMATIC PCS 7 OSx Release 4.1.2 supports the following new features:

- **OSx Merge Utility** — This utility allows you to merge the configuration from one OSx system (or a subset of that system) into another, currently running OSx system. This means that you can do major configuration development outside of an OSx system that is running a process, and then add it in without shutting down the process.
- **Remote computer data archiving** — Data archives can be stored on a remote computer. This can be any computer system that can support an FTP server; for example, UNIX, Linux, Windows NT, Windows 2000.
- **Permanent select list** — This feature allows you to choose whether the select list for tag details, graphics, reports, and so on (accessed from the Directory button) remains on the screen until you dismiss it, or disappears when you select an entry.
- **Graphic/tag cross reference report** — A standard report cross-referencing tags in graphics, by tag and by graphic, is available.
- **Internet Protocol netmask configuration** — You are prompted to specify the netmask value or select a default at installation.
- **Save new tag install file to hard disk** — The feature allows saving a tag file to hard disk in addition to MO disk and diskette.
- **SIMATIC Rack PC 840 support** — The Rack PC 840 will now be supported as a system unit.
- **1.3 gigabyte and 640 megabyte MO disk support** — Data Archiving and Backup/Restore will support larger MO disks with the Rack PC 840 hardware platform. Earlier 230 and 540 megabyte MO disks are still supported as well.
- **Additional printer support** — New printers in the Hewlett-Packard DeskJet line are supported.

Conventions Used in the Manual Set

The procedures in the various manuals give you step-by-step instructions about how to carry out tasks. Typically, the last step of any procedure requires that you select the **OK** or **Save** button, or press **Enter**. To save space and avoid redundancy, this last step does not appear in the procedure. However, you need to finish each procedure with one of these actions.

OK

OK Saves information that you have entered and closes the window.

Save

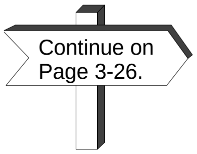
Save Saves information that you have entered and does not close the window.

Cancel

Cancel Closes the window without saving any information that you entered and terminates any action that you initiated.

Close

Dismiss Closes the window. If you have already pressed **Enter**, your work is not lost; if you have not pressed **Enter**, your work is discarded.

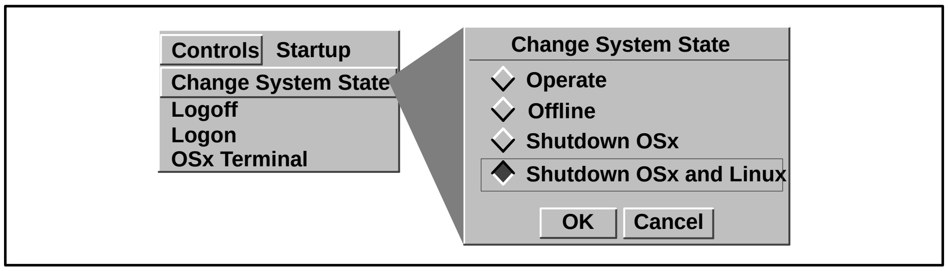


The signpost indicates that the procedure that you are currently following continues on the indicated page.

The different fonts used in the manual set have the following meanings.

- Entries that you type from the keyboard are indicated with the `courier font`.
- Items that you select on the screen, or keys that you press on the keyboard, are indicated with this **bolded font.**

Items that you select on a cascaded menu are linked in the manual text with arrows. The first term indicates where to click the main menu bar. For example, **Controls–>Change System State** tells you to click **Controls** on the main menu bar, then select **Change System State** from the pull-down menu.



Purpose of This Manual

The *SIMATIC PCS 7 OSx Interface to S7 Controllers Manual* discusses the following topics:

- Chapter 1 describes how to set up your S7 engineering station to communicate with OSx and with the S7 controllers.
- Chapter 2 describes how to create and install OSx tags from the OSx Library function blocks and symbols for S7 controllers.
- Chapter 3 describes how to create a sample S7 program with a CFC and an SFC.
- Chapter 4 describes how to create a new OSx tag type for S7 controllers.
- Appendix A shows an example of the Tag Installation report.
- Appendix B gives the simulation code for the sample S7 program.
- Appendix C gives the SCL code for a sample user-defined tag type called TANK.
- Appendix D lists error messages and describes miscellaneous troubleshooting issues for the S7 interface with OSx.

The Other Manuals

The PCS 7 OSx manual set consists of several manuals. If you cannot find the information that you need in the *SIMATIC PCS 7 OSx Interface to S7 Controllers Manual*, check these other books:

- *SIMATIC PCS 7 OSx System Administration Manual* This manual offers help in configuring network nodes, and includes procedures that describe how to configure printers, how to archive data, and how to back up files.
- *SIMATIC PCS 7 OSx Process Configuration Manual* This manual describes the primary tasks required to configure your OSx station for controlling your process.
- *SIMATIC PCS 7 OSx Graphical Editor Manual* This manual describes how to create the graphical displays used with PCS 7.
- *SIMATIC PCS 7 OSx Hardware Manual* This manual describes the various hardware components of the system and how to install them.
- *SIMATIC PCS 7 OSx Reports Manual* This manual describes how to create reports on your process and your PCS 7 configuration.
- *SIMATIC PCS 7 OSx Recipe Manual* For advanced configuration tasks involving the creation and use of recipes, refer to this manual.
- *SIMATIC PCS 7 OSx Batch Programming Manual* This manual describes configuration tasks involving the use of BCL, the Batch Control Language, and creating batch programs.
- *SIMATIC PCS 7 OSx Operator Manual* This manual describes how to carry out the various tasks that the process operator must do when the system is in the Operate state. You may photocopy all or portions of this manual as a reference for your operators.
- *SIMATIC PCS 7 OSx Interface to S5 Controllers Manual* This manual describes the OSx interface with SIMATIC S5 controllers.
- *SIMATIC PCS 7 OSx Library Manual* This manual describes function blocks used to program the S7-400 controllers to interface with OSx.

Be sure to check the Readme File for information that did not become available until after the publication deadlines for the OSx manuals. The Readme File also points to important copyright, licensing, and warranty information. Select **Help–>About OSx** from the main menu bar, and then click the **Show Readme** button at the bottom of the About OSx dialog box.

Optional PCS 7 OSx Features

The following manuals are available for optional PCS 7 OSx features.

- *SIMATIC PCS 7 OSx Remote Data Transfer Manual* This manual describes the remote data transfer feature, which allows you to transmit data collected from the process by an OSx station to an Oracle database on the remote computer for historical records and other purposes.
- *SIMATIC PCS 7 OSx X Terminal User Manual* This manual describes how to connect and operate an X terminal as an extension of an OSx station.
- *SIMATIC PCS 7 OSx @aGlance User Manual* This manual describes how to import OSx data into a Windows application, such as Excel or Lotus 1-2-3, or into another UNIX or VMS application.

If You Need Help

If you have difficulty with your system, contact the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421. Outside the U.S.A., call 49-911-895-7000.

Chapter 1 Getting Started

1.1 Introduction

Overview

SIMATIC PCS 7 OSx now supports the S7-400 family of controllers. This manual is intended for the STEP 7 control engineer to incorporate OSx library blocks in the S7 controller program to be used as OSx tags. This manual describes the requirements for OSx to support S7 controllers. The standard Engineering Toolset along with add-ons allows the OSx to communicate with these controllers. A special library has been created for use to transfer blocks as tags to OSx. A mapper utility has been developed to take the function blocks and symbols and store them in a file to be used to install tags on OSx. A utility has been developed to transfer this file to the OSx for tag installation.

This manual assumes that the SIMATIC PCS 7 Engineering Toolset with the SIMATIC Manager, STEP 7, SFC, CFC, and SCL is already installed on the engineering station. For the S7-417H controller, the Engineering Toolset must be Version 5.0 or later. A list of required tools that need to be added are described along with the installation procedure.

The required firmware for proper communication is listed on page 1-12. This chapter explains how to set up the communication link between OSx and S7 controllers. SOFTNET S7 licensing is required for this link and ethernet address configuration.

This manual describes how you use the library and the utilities to connect the OSx and the S7 controller.

Installing the Engineering Toolset

Before you can begin development of your S7 program for OSx connection, you must install the Engineering Toolset from the CD-ROM labeled *Process Control System PCS 7 Toolset*. Check to see which version of the Toolset you are using, and follow the appropriate procedure for your version of the Engineering Toolset:

- Version 4.01 — page 1-4
- Version 4.02 — page 1-5
- Version 5.1 — page 1-6
- Version 5.2 — page 1-8

Installing Version 4.01

If you are using version 4.01 of the Engineering Toolset, you need to load the following programs from the CD-ROM labeled *SIMATIC PCS 7 OSx Library* (Order No. 6ES7 653-1BX01-0XX0):

- SIMATIC AS-OS-Engineering V 4.01 — This component includes the extensions that allow you to transfer configuration data from the engineering station to both OS and OSx stations.
- SIMATIC AS-OS-Engineering OSx V 4.01 — This component includes the mapper utility that is used to translate function blocks and symbols into OSx tags.
- SIMATIC OSx Library — This component includes the function block library that allows you to program S7-400 controllers to interface with the OSx system.
- SIMATIC STEP 7-OSx Transfer Utility — This component is the utility that is used to transfer OSx tag files from a STEP 7 engineering station to an OSx station.

If you installed SIMATIC AS-OS-Engineering V 4.01 when you installed the Engineering Toolset, you must uninstall it by following the steps below:

1. From the **Start** button on the Windows taskbar, select **Settings->Control Panel**. The Control Panel appears.
2. Double-click the **Add/Remove Programs** icon. The Add/Remove Program Properties dialog box appears.
3. In the Install/Uninstall folder, double-click on **SIMATIC AS-OS Engineering OS V 4.01**. Click **Yes** at the prompt. The program is removed.
4. Click **OK**.

You are now ready to install the required programs. Follow the steps below:

1. Insert the OSx Library CD-ROM in the drive.
2. From the **Start** button on the Windows taskbar, select **Run**. The Run dialog box appears.
3. Enter `x:\setup.exe`, where `x` is the CD-ROM drive. Click **OK**, and follow the prompts until you reach the Select Components dialog box. Make sure that all four required programs are checked.
4. Click **Next** to start loading the programs and respond appropriately to the prompts until all the required programs are loaded.

Installing Version 4.02

If you are using version 4.02 (SP1) of the Engineering Toolset, you need to load the following programs from the CD-ROM labeled *SIMATIC PCS 7 OSx Library* (Order No. 6ES7 653-1BX02-0XX0):

- SIMATIC AS-OS-Engineering OSx V 4.02 — This component includes the mapper utility that is used to translate function blocks and symbols into OSx tags.
- SIMATIC OSx Library V 4.1.0 — This component includes the function block library that allows you to program S7-400 controllers to interface with the OSx system.
- SIMATIC STEP 7-OSx Transfer Utility — This component is the utility that is used to transfer OSx tag files from a STEP 7 engineering station to an OSx station.

To install the required program, follow the steps below:

1. Insert the OSx Library CD-ROM in the drive.
2. From the **Start** button on the Windows taskbar, select **Run**. The Run dialog box appears.
3. Enter `x:\setup.exe`, where `x` is the CD-ROM drive. Click **OK**, and follow the prompts until you reach the Select Components dialog box.
4. Three programs are listed in the Select Components dialog box. Make sure all three boxes are checked.
5. Click **Next** to start loading the programs and respond appropriately to the prompts until all three required programs are loaded.

Installing Version 5.1

If you are using version 5.1 of the Engineering Toolset, you need to load the following programs:

- SIMATIC AS-OS-Engineering — This component includes the extensions that allow you to transfer configuration data from the engineering station to both OS and OSx stations.

  One of the two following versions of the component must be installed before you install the SIMATIC AS-OS-Engineering **OSx** V 4.02 below:

  If WinCC is installed: SIMATIC AS-OS-Engineering V 5.2 from the Engineering Toolset CD enables tag transfer for both WinCC and OSx.

  NOTE: The SIMATIC AS-OS Engineering V 5.2 program does not install unless WinCC is already installed.

  If WinCC is not installed: In this case, the Engineering Toolset is being used only with the OSx host system.

  To enable tag transfer to OSx, use the OSx Library CD to install the SIMATIC AS-OS-Engineering V 4.02 program.

  You can use the Windows utility Add/Remove Programs in the Control Panel to determine if some version of SIMATIC AS-OS Engineering is already installed. If so, test the tag transfer operation before removing that version and installing the V4.02 version.

- SIMATIC AS-OS-Engineering OSx V 4.02 — This component includes the mapper utility that is used to translate function blocks and symbols into OSx tags.
- SIMATIC OSx Library V 4.1.1 — This component includes the function block library that allows you to program S7-400 controllers to interface with the OSx system.
- SIMATIC STEP 7–OSx Transfer Utility — This component is the utility that is used to transfer OSx tag files from a STEP 7 engineering station to an OSx station.

To install the required programs, follow the steps below:

1. If WinCC is installed on your system, insert the PCS 7 Engineering Toolset CD in the drive, and load SIMATIC AS-OS-Engineering V 5.2. (WinCC must be installed first.) Remove the CD, and continue with step 2.

   If you do not plan to use WinCC, begin with step 2.

2. Insert the OSx Library CD in the drive.

3. From the **Start** button on the Windows taskbar, select **Run**. The Run dialog box appears.

4. Enter `x:\setup.exe`, where `x` is the CD-ROM drive. Click **OK**, and follow the prompts until you reach the Select Components dialog box.

5. Four programs are listed in the Select Components dialog box. If WinCC is not installed on your system, check the box next to SIMATIC AS-OS Engineering V 4.02. Otherwise, leave it blank.

   Make sure the other three boxes are checked, including SIMATIC AS-OS-Engineering **OSx** V 4.02.

6. Click **Next** to start loading the programs and respond appropriately to the prompts until all required programs are loaded.

Installing Version 5.2

If you are using version 5.2 of the Engineering Toolset, you need to load the following programs from the CD-ROM labeled *SIMATIC PCS 7 OSx Library* (Order No. 6ES7 653-1BX02-0XX0):

- SIMATIC AS-OS-Engineering — This component includes the extensions that allow you to transfer configuration data from the engineering station to both OS and OSx stations.

  One of the two following versions of the component must be installed before you install the SIMATIC AS-OS-Engineering **OSx** V 4.02 below:

  If WinCC is installed: SIMATIC AS-OS-Engineering V 5.2 including SP1 from the Engineering Toolset CD enables tag transfer for both WinCC and OSx.

  NOTE: The SIMATIC AS-OS Engineering V 5.2 program does not install unless WinCC is already installed.

  If WinCC is not installed: In this case, the Engineering Toolset is being used only with the OSx host system.

  To enable tag transfer to OSx, use the OSx Library CD to install the SIMATIC AS-OS-Engineering V 4.02 program.

  You can use the Windows utility Add/Remove Programs in the Control Panel to determine if some version of SIMATIC AS-OS Engineering is already installed. If so, test the tag transfer operation before removing that version and installing the V4.02 version.

- SIMATIC AS-OS-Engineering OSx V 4.02 — This component includes the mapper utility that is used to translate function blocks and symbols into OSx tags.

- SIMATIC OSx Library V 4.1.1 — This component includes the function block library that allows you to program S7-400 controllers to interface with the OSx system.

- SIMATIC STEP 7-OSx Transfer Utility — This component is the utility that is used to transfer OSx tag files from a STEP 7 engineering station to an OSx station.

To install the required programs, follow the steps below:

1. If WinCC is installed on your system, insert the PCS 7 Engineering Toolset CD in the drive, and load SIMATIC AS-OS-Engineering V 5.2 including SP1. (WinCC must be installed first.) Remove the CD, and continue with step 2.

   If you do not plan to use WinCC, begin with step 2.

2. Insert the OSx Library CD in the drive.

3. From the **Start** button on the Windows taskbar, select **Run**. The Run dialog box appears.

4. Enter `x:\setup.exe`, where `x` is the CD-ROM drive. Click **OK**, and follow the prompts until you reach the Select Components dialog box.

5. Four programs are listed in the Select Components dialog box. If WinCC is not installed on your system, check the box next to SIMATIC AS-OS Engineering V 4.02. Otherwise, leave it blank.

   Make sure the other three boxes are checked, including SIMATIC AS-OS-Engineering **OSx** V 4.02.

6. Click **Next** to start loading the programs and respond appropriately to the prompts until all required programs are loaded.

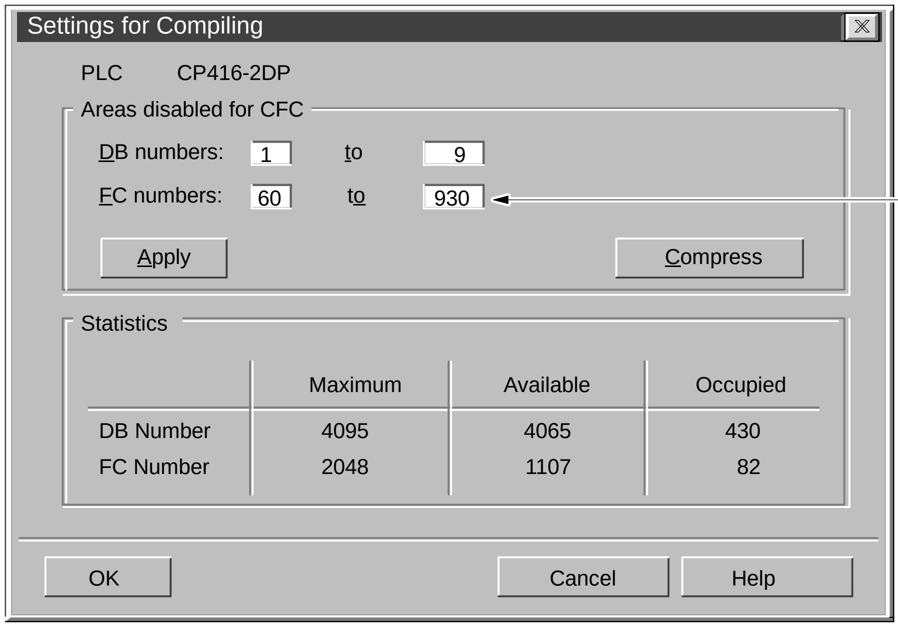
Reserving Memory for OSx Functions

Function block numbers in the OSx library range from FB340 to FB407; functions range from FC900 to FC929. The default range for functions in CFCs does not cover the OSx functions. To ensure proper operation of the CFC with the OSx functions, you must reset the default for FC numbers. Follow the steps below:

1. Open any CFC in an S7 program.
2. Select **Options–>Compilation Settings** from the menu bar. The Settings for Compiling dialog box appears.
3. Under the heading Areas Disabled for CFC, change the higher of the FC numbers to 930 or greater (Figure 1-1).
4. Select **OK**. The appropriate memory locations are now reserved for the OSx library functions.

The new default setting applies to the CFCs in all of your S7 programs. You do not have to reset it separately for each CFC.

In an S7-417 controller, the total number of data blocks (DBs) available is 8191. In an S7-416 controller, the total number of data blocks (DBs) available is 4095; in an S7-414 controller, the number is 1023. If you create a data block, make sure that the range of disabled DB numbers in this dialog box includes the number of the data block. For example, if you create a data block DB12, the DB Numbers field must read 1 to at least 12.


Settings for Compiling

PLC CP416-2DP

Areas disabled for CFC

DB numbers: 1 to 9

FC numbers: 60 to 930

Statistics

	Maximum	Available	Occupied
DB Number	4095	4065	430
FC Number	2048	1107	82

Change the value in this box to 930.

Figure 1-1 Settings for Compiling Dialog Box

1.2 Required Firmware

When you use S7 controllers and the Engineering Toolset with SIMATIC PCS 7 OSx, it is recommended that you always use the latest version of firmware for CPUs, communications cards, and other components. If you have questions about the compatibility of firmware components, contact the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421. Outside the U.S.A., call 49-911-895-7000.

To check the version number on your CPU firmware, follow the steps below.

1. Open the SIMATIC Manager.
2. Make sure that you are in the Standard Hierarchy view. If not, select **View->Standard**.
3. Click on the CPU to highlight it.
4. Select **PLC->Module Information**. The version number appears just below the order number.

To check the version number on your CP443-1 firmware, follow the steps below.

1. Open the SIMATIC Manager.
2. Make sure that you are in the Standard Hierarchy view. If not, select **View->Standard**.
3. Click the right mouse button on the CP443-1 and select **Object Properties**. The Block Object Properties dialog box appears.
4. Click the **Diagnostics** tab.
5. Click the **Run** button. The version number appears just below the CP type.

1.3 RBE Considerations

Alarm_8 Restrictions

The S7-417, S7-416 and S7-414 controllers limit the number of hosts that can sign on for the S7 messaging services, Alarm_S and Alarm_8. OSx uses these services for Report By Exception (RBE) messages.

The S7-417 controller and the original models of the S7-416 controller allow up to eight OSx stations. The original models of the S7-414 controller allow up to four OSx stations. Newer models that support more OSx stations are listed in Table 1-1 and Table 1-2.

If all the S7-414 controllers in your system are from a release listed in Table 1-1 (or a newer model S7-414), you can connect up to eight OSx stations.

Table 1-1 S7-414 Controllers That Support Eight OSx Stations

Order Number	Release
6ES7-414-1XG01-0AB0	A5
6ES7-414-2XG01-0AB0	A5
6ES7-414-2XJ01-0AB0	A7
6ES7-414-1XG02-0AB0	A3
6ES7-414-2XG02-0AB0	A3
6ES7-414-2XJ02-0AB0	A3
6ES7-414-3XJ00-0AB0	ALL

If all the S7-416 controllers in your system are from a release listed in Table 1-2 (or a newer model S7-416), you can connect up to 12 OSx stations.

Table 1-2 S7-416 Controllers That Support 12 OSx Stations

Order Number	Release
6ES7-416-1XJ01-0AB0	A5
6ES7-416-2XK00-0AB0	A7
6ES7-416-2XL00-0AB0	A7
6ES7-416-1XJ02-0AB0	A3
6ES7-416-2XK01-0AB0	A2
6ES7-416-2XL01-0AB0	A2
6ES7-416-3XL00-0AB0	ALL

In order to take advantage of the added capacity of the new S7 controller releases, you must run a command from the UNIX command line to change the default number of OSx stations. Follow these steps.

1. Select **Controls->Change System State** from the main menu bar.
2. Select **Offline** to set OSx to the Offline state.
3. Select **Controls->OSx Terminal** from the main menu bar.
4. Log in as `tistar`.
5. At the prompt, type `s7limit.sh new` and press **Enter**. You can now connect up to the new maximum of OSx stations to your system.
6. Type `exit` to exit the OSx Terminal session.

If for some reason you need to restore the old limits of four OSx stations in a system with S7-414 controllers, or eight in a system with S7-416 or 417 controllers, follow the procedure above but type `s7limit.sh old` in step 5.

The presence of any other device that can sign on to the S7 controller also reduces the number of OSx stations that can sign on for Alarm_8 and Alarm_S messaging. For example, suppose that an S7 programming device signs on to an S7-416 controller using the **PLC–>CPU Messages** command, while an eight-station OSx system is in the Offline state. Now there are only seven remaining slots available for sign-on. When the system transitions to the Operate state, none of the eight OSx stations signs on for RBE processing, since all OSx stations must have identical RBE information. The S_CNODE report indicates **NORBE**. After the S7 programming device signs off and eight slots are again available, a **Scan_On** command for the S7-416 controller issued from Network Setup establishes the RBE connection for all eight OSx stations.

Alternatively, if a system with eight OSx stations is already in the Operate state, an S7 programming device will not be allowed to sign on to the S7-416, unless you select the **Scan_Off** command from Network Setup first, turning off RBE delivery for OSx until the programming device signs off. Another solution is to set one of the OSx stations to out_of_service, freeing up one sign-on slot for the programming device.

To avoid these situations altogether, the effective limit on the number of OSx stations in a system with S7 controllers must be reduced by the number of other devices that can sign on to the controllers. For example, in a system with models of S7-416 controllers that support eight sign-ons, an arrangement of seven OSx stations with one CP443-1 and one S7 programming device allows all components to sign on at the same time with full RBE messaging capability.

RBE Considerations (continued)

RBE Comparison

In the S7 controllers: RBE is implemented within the special OSx library of blocks. These “tag” objects internally call an RBE Adapter function block (for either ALARM_S or ALARM_8P) which manages the message delivery in a manner to conform to the OSx RBE requirements. When an object from this library is assigned to an OSx tag, then the tag delivers an RBE message if it is used by OSx for RBE (such as an alarm tag or an autolog tag).

The S7 controller programs require the OSx library blocks to provide the RBE messages.

In the S7 controller, the CPU generates messages from any block programmed to have a message. If only the OSx library objects have been used in the program, then the ALARM_S or ALARM_8P messages conform to OSx functionality. Messages coming from any non-OSx library block are considered non-OSx messages. However, since OSx receives all configured messages, any non-OSx messages are ignored, but may still reduce the performance of the OSx connection.

RBE Memory Allocation

Earlier models of the S7-414 and 416 have fixed ALARM_8P messaging capabilities. For the 414 model, the capacity is 300 messages. For the 416 model, the capacity is 600 messages.

For newer versions of the 414-2 and all versions of the 414-3 models, you can configure the message capacity in the SIMATIC Manager under **Object Properties->Memory** up to a maximum of 600 messages; the default is 300. For newer versions of the 416-2 and all versions of the 416-3 models, you can configure the message capacity up to a maximum of 1800 messages; the default is 600. For the 417 model, you can configure the message capacity up to a maximum of 10,000 messages; the default is 1200.

The function blocks (tag types) of AI and LOOP from the OSx Library each use one RBE message. The message type for AI and LOOP is ALARM_8P, which is described in more detail in the STEP 7 documentation. With an S7-416 controller, the program can hold up to a maximum of 600 AI and/or LOOP tags (provided that other non-OSx function blocks have not used any ALARM_8P messages).

The other S7 message type used by OSx is the ALARM_S message. The ALARM_S message is managed separately and does not interact with the fixed message capacity. Instead, the ALARM_S interacts with a separate fixed capacity of 200 ALARM_S messages that are simultaneously active. That is, it is possible to have many more than 200 ALARM_S messages configured, but only 200 can be in the active state. The OSx RBE implementation provides for automatic management of the ALARM_S message delivery to reduce or eliminate the possibility of ever reaching the situation of having 200 ALARM_S messages remaining in an active state.

With the OSx management of ALARM_S messages, you can have hundreds of ALARM_S messages in an S7 program. All device and discrete function blocks in the OSx Library provide ALARM_S message capability.

For the S7, the program holds the message configuration. Each S7 CPU has a predefined message capacity. The S7-416 CPU can hold twice as many messages as the S7-414 CPU. The message type of ALARM_8P is part of the fixed message resource. OSx uses this message type for AI and LOOP tag types. The other OSx tag types use a message type of ALARM_S. The ALARM_S type is dynamically allocated and is limited to a certain number active at any time (200). Within OSx, checks are made for each OSx tag type having an active ALARM_S, and if necessary, an RBE command is issued so that an active message is freed. This free command merely frees an entry in the resource; the true tracking of the RBE signal state is not affected. In summary, the possibility of reaching the S7 memory resource is minimized under OSx since it is reached only when a large number of AI or LOOP tags are assigned.

1.4 Communications Requirements

S7 Communication

The S7 protocol is used for data transmission between these pairs of network nodes:

- OSx station and the S7 control nodes
- An S7 engineering station and the S7 control nodes

The S7 protocol requires a valid license for SOFTNET-S7 on each OSx station. You need the Engineering Toolset software on the engineering station to provide the S7 protocol.

TCP/IP

For communication between OSx stations and between an OSx station and the management information service (MIS) computer, the TCP/IP protocol is used. This protocol locates these stations through their respective Internet protocol (IP) addresses. This is the IP address that you assign during initialization of an OSx station. See the *PCS 7 OSx System Administration Manual* for more information.

The TCP/IP protocol is also used to transfer the mapper file for tag installation from the S7 engineering station to OSx.

Order of Tasks

Once you have the required releases of the system hardware, you need to configure all the units to communicate with one another. To configure communications between an S7 controller, an engineering station, and an OSx station, you must perform the following tasks:

- Configure the engineering station (Section 1.5)
- Configure the S7 control node on OSx (Section 1.6)
- Configure the CP 443-1 communications processor (Section 1.7)

1.5 Configuring the Engineering Station

The Engineering Station

An S7 engineering station is equipped with the Engineering Toolset, Version 4.01 (Order Number 6ES7 651-8XX01-0YD0) to support programming the S7 controllers. The Engineering Toolset includes STEP 7, SCL, SFC, CFC, and AS-OS Engineering Tag Mapper. The S7 controllers also require the Engineering Toolset OSx Library to handle RBE messaging for OSx (page 1-4 and following).

For a connection to the S7 controller, the Engineering Toolset offers the following options:

- An MPI cable between the serial comm port and the MPI interface in the S7 controller
- A special MPI card in the engineering station, which provides a faster direct connection
- An Industrial Ethernet network connection to the CP443-1 card in the S7 controller, which requires a supported driver and hardware interface

After an S7 engineering station is connected to the network, you can use this connection to load the application program to the appropriate control node. If available, you can also use the Industrial Ethernet network connection to transfer mapper files to OSx for tag installation.

You do not need to configure the engineering station in OSx Network Setup in order to transfer the mapper files. The TCP/IP protocol is used to transfer mapper files. For the station with the Engineering Toolset, the IP address must be configured using the standard Windows procedure for network setup. The IP address for each ethernet card on a Windows NT platform must be configured on a unique subnet.

1.6 Configuring the S7 Control Node on OSx

Licensing

Verify that you have a valid license for the SOFTNET-S7. OSx permits configuration of all supported controllers. However, a valid SOFTNET-S7 license is required in order for OSx to communicate to the S7 controllers.

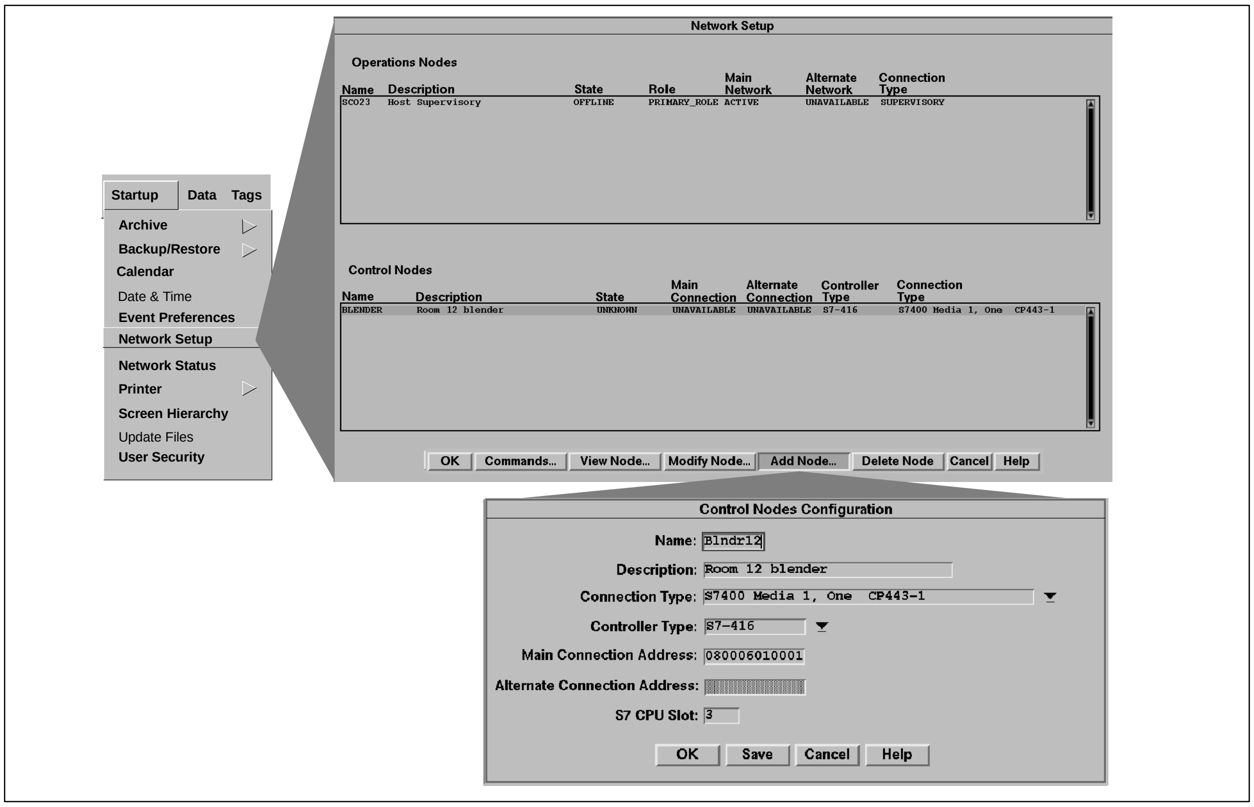
To verify your license, select **Help–>About OSx** on the OSx station. License information appears in the Options Installed field. Be sure that this field says **Valid S7 License Present**.

Configuring an S7 Controller

Figure 1-2 shows the Control Nodes Configuration dialog box for an S7 controller. Note the absence of the RBE Delivery field and the presence of the S7 CPU Slot field, showing the slot in which the S7 CPU resides. The default is 3.

In the S7 control node, each RBE message is transmitted separately at the earliest opportunity. Therefore, in the S7 RBE implementation, the RBE Delivery setting does not apply.

The dual connection type provides greater responsiveness for operator commands and data collection compared to a single connection. For an S7 control node, a dual connection type is recommended, even when only one CP443–1 is present. For an S7 controller with two CP443-1 modules, you can configure either a dual or a redundant connection. See the *SIMATIC PCS 7 OSx System Administration Manual* for further information on connection types.



Operations Nodes

Name	Description	State	Role	Main Network	Alternate Network	Connection Type
SCO23	Host Supervisory	OFFLINE	PRIMARY_ROLE	ACTIVE	UNAVAILABLE	SUPERVISORY

Control Nodes

Name	Description	State	Main Connection	Alternate Connection	Controller Type	Connection Type
BLENDER	Room 12 blender	UNKNOWN	UNAVAILABLE	UNAVAILABLE	S7-416	S7400 Media 1, One CP443-1

Name:	Blndr12
Description:	Room 12 blender
Connection Type:	S7400 Media 1, One CP443-1
Controller Type:	S7-416
Main Connection Address:	080006010001
Alternate Connection Address:	
S7 CPU Slot:	3

Figure 1-2 Adding an S7 Control Node to the System

1.7 Configuring the CP443-1 for S7 Controllers

Using the S_CPCFG Report for CP Configuration

For detailed instructions about how to configure a CP443-1 card, refer to the documentation that accompanies the STEP 7 software. A general procedure is listed below.

NOTE: These general guidelines do not replace the detailed procedures in the documentation that accompanies the STEP 7 software. Consult the STEP 7 documentation for configuring the CP443-1.

1. Do the control node configuration from an OSx station, as described in the *SIMATIC PCS 7 OSx System Administration Manual*.
2. Run the standard report **S_CPCFG** on the OSx primary that has the control node configuration and print the report. The information contained in the report is described briefly on the next page. A report example is shown in Figure 1-3.
3. Refer to the report and use the STEP 7 hardware configuration utility to configure the CP443-1 card for each S7 control node.

Figure 1-3 shows an **S_CPCFG** report for two S7 controllers called LINE_1 and LINE_2. To the right of each control node name, the information for that control node appears in the following order: description, connection type, ethernet address(es), and total connections required.

The ethernet address listed in the report should be the same as the ethernet address entered in the MAC Address field of the STEP 7 hardware configuration as shown in Figure 1-4. For a dual connection using a single CP443-1, such as the LINE_2 controller in the example, you enter the ethernet address in STEP 7 only once.

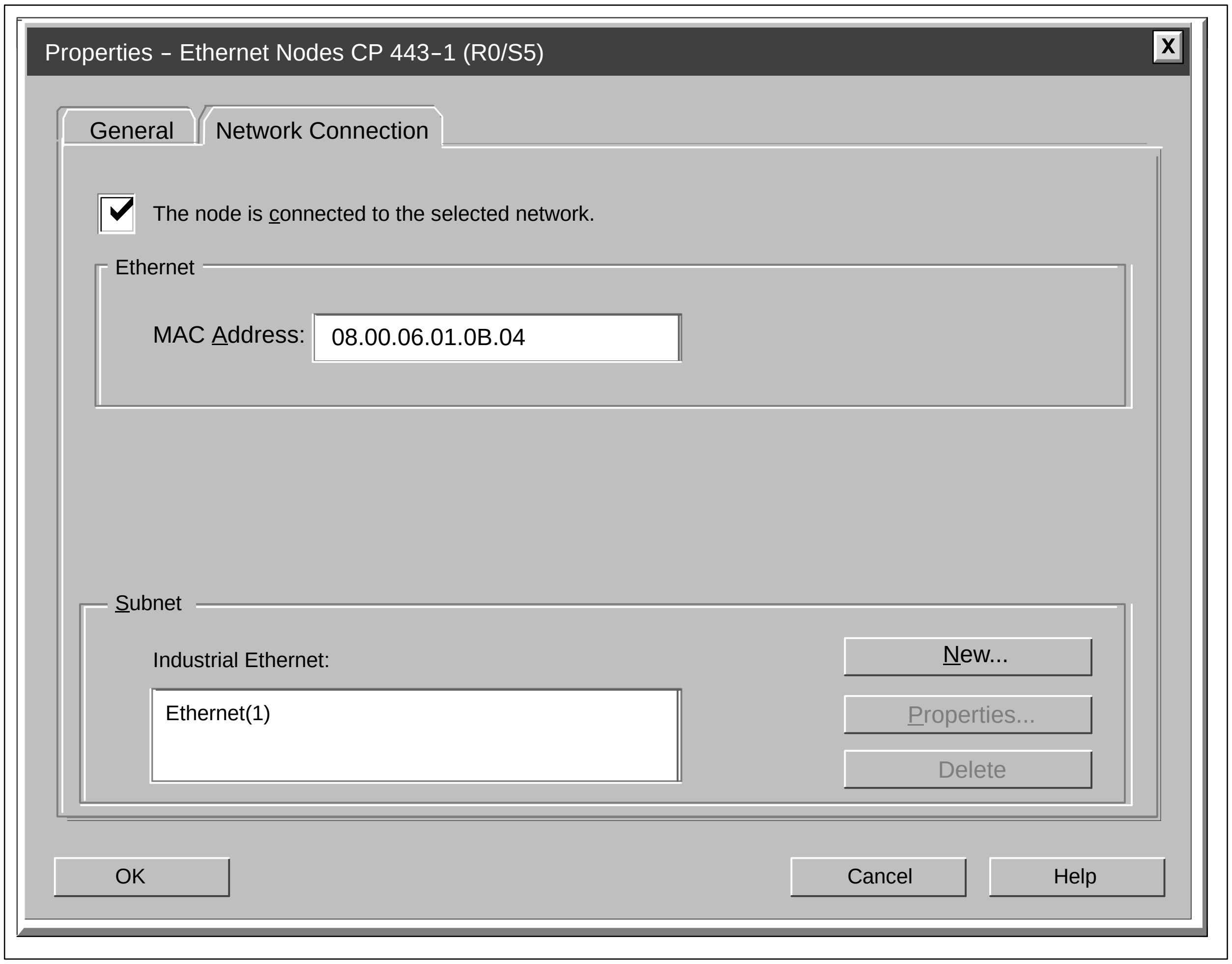
```
Control Node Report                    Page 1                         31-August-98

LINE_1    : S7 controller for line 1
           416-2DP S7400 Media 1, One CP443-1
           Media 1 (main) Ethernet address = 080006010B04
           Total connections required by PCS 7 OSx = 1 in LINE_1


LINE_2    : S7 controller for line 2
           416-2DP S7400 Media 1, Dual CP443-1
           Media 1 (main) Ethernet address = 080006010B05
           Media 1 (alt) Ethernet address = 080006010B05
           Total connections required by PCS 7 OSx = 2 in LINE_2
```

Figure 1-3 S_CPCFG Report for S7 with Single and Dual Connections

Configuring the CP443–1

For the CP443–1 module, the assignment of the ethernet address is the only configuration required for OSx since the S7 protocol provides a mechanism used by OSx to provide automatic connection assignment. For S7 controllers, this information is stored in the S7 CPU and in the CP443–1 module. Do this configuration within STEP 7.

When you select the CP443-1 module from the Hardware Catalog during S7 hardware configuration, the Properties screen in Figure 1-4 appears. Assign the CP443–1 module the ethernet address on the preprinted label that came with the module. Enter this address in the MAC Address field.



✓ The node is connected to the selected network.

Ethernet

MAC Address: 08.00.06.01.0B.04

Subnet

Industrial Ethernet: Ethernet(1)

Figure 1-4 STEP 7 CP443–1 Hardware Configuration

Chapter 2
Configuring and Installing S7 Tags

2.1 Configuring S7 Blocks and Symbols for OSx

Overview

Function blocks in the OSx library are intended for use with the SIMATIC PCS 7 OSx based system. The function blocks are optimized as OSx tags to meet OSx alarm messaging requirements. In addition, these blocks are designed for direct use with the OSx graphical interface. Other function blocks provided by STEP 7 and the Engineering Toolset, such as the Library of Standard Blocks or the Library of Technogical Blocks, can also be used in an S7 application for OSx, but are not optimized for the interface to OSx and cannot be used for OSx alarm messaging or with the standard graphical objects provided in OSx, such as valves, motors, faceplates, and tag details.

Guidelines for Programming S7 Controllers for OSx

When you program an S7 controller for OSx, you use the SIMATIC PCS 7 Engineering System software, referred to as the Engineering Toolset. The name of the S7 program name must be identical to the control node name as configured on the OSx. For detailed information about programming in STEP 7, refer to the STEP 7 user documentation.

Any blocks that interface with the OSx environment (in graphics, reports, alarms, recipes), must come from the OSx Library in the Engineering Toolset. These blocks have been specifically configured to function as tags in OSx. You can also configure STEP 7 symbol names to be used for selected types of OSx tags.

Function blocks and symbols are marked differently for transfer to OSx. See Section 2.2 for configuring parameters for function blocks; see Section 2.3 for configuring parameters for symbols. For more information about configuring OSx tags, see the *SIMATIC PCS 7 OSx Process Configuration Manual.*

In STEP 7, all names are case sensitive. In OSx, the tag names are case insensitive. For example, the names "Tag1" and "tag1" can represent two different names within STEP 7, but indicate the same name in OSx. To avoid confusion, it is recommended that all names used for OSx tags be kept in lower case in STEP 7.

2.2 Creating Tags from Function Blocks

List of Blocks

Table 2-1 shows an alphabetical listing of all the function blocks and functions in the OSx Library of Blocks for the Engineering Toolset, and the OSx tag types that they correspond to, if applicable. See the *SIMATIC PCS 7 OSx Library Manual* for detailed descriptions of these blocks.

Table 2-1 OSx Library of Blocks

Name	Description	Block Number	OSx Tag Type
ABS_MTH	Absolute value	FC900	---
AI	Analog Input	FB440	calc
ALRM	Analog alarm	FB384	ai
AO	Analog output	FB406	ao
AREA	Area	FB404	area
ARWPC	Anti-reset windup protection / constraint type	FB427	do
ARWPS	Anti-reset windup protection / select type	FB428	do
AVG_SEL	Average selector	FB375	---
BCDBIN	BCD to binary conversion	FB409	---
BINBCD	Binary to BCD conversion	FB410	---
BI	BCD Input	FB432	ivar
BIT_ASGN	Bit assign	FC912	---
BITCLEAR	Bit clear	FC913	---
BITSET	Bit set	FC914	---
BITS_INT	Bits to integer	FC915	---
BITTEST	Bit test	FC916	---
BO	BCD Output	FB433	ivar
BV1	Three-position valve/type 1	FB351	mtr2
BV2	Three-position valve/type 2	FB352	mtr2
CALC	Calculated value	FB402	calc
CORLT	Correlated Lookup Table	FB439	do
CSD	Single-drive/dual-feedback cylinder	FB371	vlv2
CT_DECL	Counter	FB340	ctr
Table continues on next page.			

Table 2-1 OSx Library of Blocks (continued)

Name	Description	Block Number	OSx Tag Type
CUD	User-defined cylinder	FB372	vlv2
DERV	Derivative	FB421	do
DI	Digital input	FB398	di
DI10	Digital input array of size 10	FB400	di10
DIV_MTH	Divider	FC901	---
DMD	Dual Mode	FB414	loop
DO	Digital output	FB399	do
DO10	Digital output array of size 10	FB401	do10
DTC	Dead Time Compensator	FB413	loop
DTD	Dead Time Delay	FB420	do
EDGE	Edge	FB388	---
FFOA	Feedforward Output Adjust	FB415	loop
FFSA	Feedforward Setpoint Adjust	FB416	loop
FLAG	Flag	FB436	di
FOLAG	First Order Lag	FB423	do
FOLL	First Order Lead Lag	FB424	do
FRAC	Fraction	FC917	---
HIGH_SEL	High selector	FB376	---
INT_BITS	Integer to bits	FC919	---
INTEG	Integrator	FB422	do
INTERPOL	Interpolate	FC918	---
ISWT_SEL	Inswitch selector	FB377	---
IVAR	Integer variable	FB403	ivar
LEAD_LAG	Lead lag	FB389	---
LEFT_SH	Left shift	FC920	---
LIMIT	Limiter	FC921	---
LOAD_ARR	Load real array	FC922	---

Table continues on next page.

Table 2-1 OSx Library of Blocks (continued)

Name	Description	Block Number	OSx Tag Type
LOAD_IAR	Load integer array	FC931	---
LOOKUP	Lookup table	FC923	---
LOW_SEL	Low selector	FB378	---
MAX	Maximum value	FB390	---
MDN	Dual-drive/null-feedback motor	FB364	mtr1
MDS	Dual-drive/single-feedback motor	FB365	mtr1
MED_SEL	Median selector	FB379	---
MIN	Minimum value	FB391	---
MINMAX	Minimum and maximum value	FB392	---
MLT_MTH	Multiplier	FC902	---
MPC	Motor Position Control	FB418	do
MSN	Single-drive/null-feedback motor	FB362	mtr1
MSS	Single-drive/single-feedback motor	FB363	mtr1
MUD	User-defined motor	FB370	mtr1
ONOFF	On/off	FB383	loop
OSWT_SEL	Outswitch selector	FB380	---
OUT_LIM	Output limiter	FB373	---
PACKBITS	Pack bits	FC924	---
PDD	Dual-drive/dual-feedback press	FB357	vlv2
PID	Proportional-integral-derivative loop	FB382	loop
PMD	Motor-drive/dual-feedback press	FB358	vlv2
PND	Hand-operated/dual-feedback press	FB355	vlv2
PS1	Three-position press/type 1	FB360	mtr2
PS2	Three-position press/type 2	FB361	mtr2
PSD	Single-drive/dual-feedback press	FB354	vlv2
PSN	Single-drive/null-feedback press	FB356	vlv1
PSS	Single-drive/single-feedback press	FB353	vlv1

Table continues on next page.

Table 2-1 OSx Library of Blocks (continued)

Name	Description	Block Number	OSx Tag Type
PTC	Proportional time control	FB385	---
PUD	User-defined press	FB359	vlv2
RATE_LIM	Rate limiter	FB374	---
RATIO	Ratio	FB417	loop
RIGHT_SH	Right shift	FC928	---
RM1	Reversible motor/type 1	FB366	rmtr
RM2	Reversible motor/type 2	FB367	rmtr
ROUND	Round	FC926	---
RTD	Resistive Temperature Detector	FB434	calc
SCL_BLK	Scale	FB387	---
SEQ_ARY	Sequence array	FB342	---
SI	Scaled Integer	FB	ao
SOLAG	Second Order Lag	FB425	do
SOLL	Second Order Lead Lag	FB426	do
SPL_RNG	Split range	FB386	---
SQR_MTH	Square	FC904	---
SQRT_MTH	Square root	FC905	---
SR_ARY	Shift register array	FB343	---
SUB_MTH	Subtractor	FC906	---
SUM_MATH	Summer	FC907	---
TA	Text Array	FB412	---
TC	Thermocouple	FB435	calc
TEXT	Text	FB407	text
THR_SEL	Threshold selector	FB381	---
TI_DECL	Timer declaration	FB341	ivar
TMR	Stopwatch Timer	FB429	---
TRUNC	Truncate	FC927	---

Table continues on next page.

Table 2-1   OSx Library of Blocks (continued)

Name	Description	Block Number	OSx Tag Type
TS1	Two-speed motor/type 1	FB368	mtr2
TS2	Two-speed motor/type 2	FB369	mtr2
UNIT	Unit	FB405	unit
UNPKBIT	Unpack bits	FC929	---
VDD	Dual-drive/dual-feedback valve	FB348	vlv2
VLV_SEQ	Valve Sequence Control	FB419	do
VMD	Motor-drive/dual-feedback valve	FB349	vlv2
VND	Hand-operated/dual-feedback valve	FB346	vlv2
VSD	Single-drive/dual-feedback valve	FB345	vlv2
VSN	Single-drive/null-feedback valve	FB347	vlv1
VSS	Single-drive/single-feedback valve	FB344	vlv1
VUD	User-defined valve	FB350	vlv2
WI	Word Input	FB430	ivar
WO	Word Output	FB431	ao

Creating Tags from Function Blocks (continued)

Marking Function Blocks

When you create an S7 program, you move blocks that you want to use as tags in OSx from the OSx Library into a CFC, where you configure them and link them to other parts of the S7 program. You then need to mark these function blocks before you can transfer them to the OSx database. The name of the S7 program name must be identical to the control node name as configured on the OSx.

NOTE: When you place a function block in a CFC, the block is assigned a default tag name, which is a numerical value. You must change this value to its OSx tag name, which cannot be just a number.

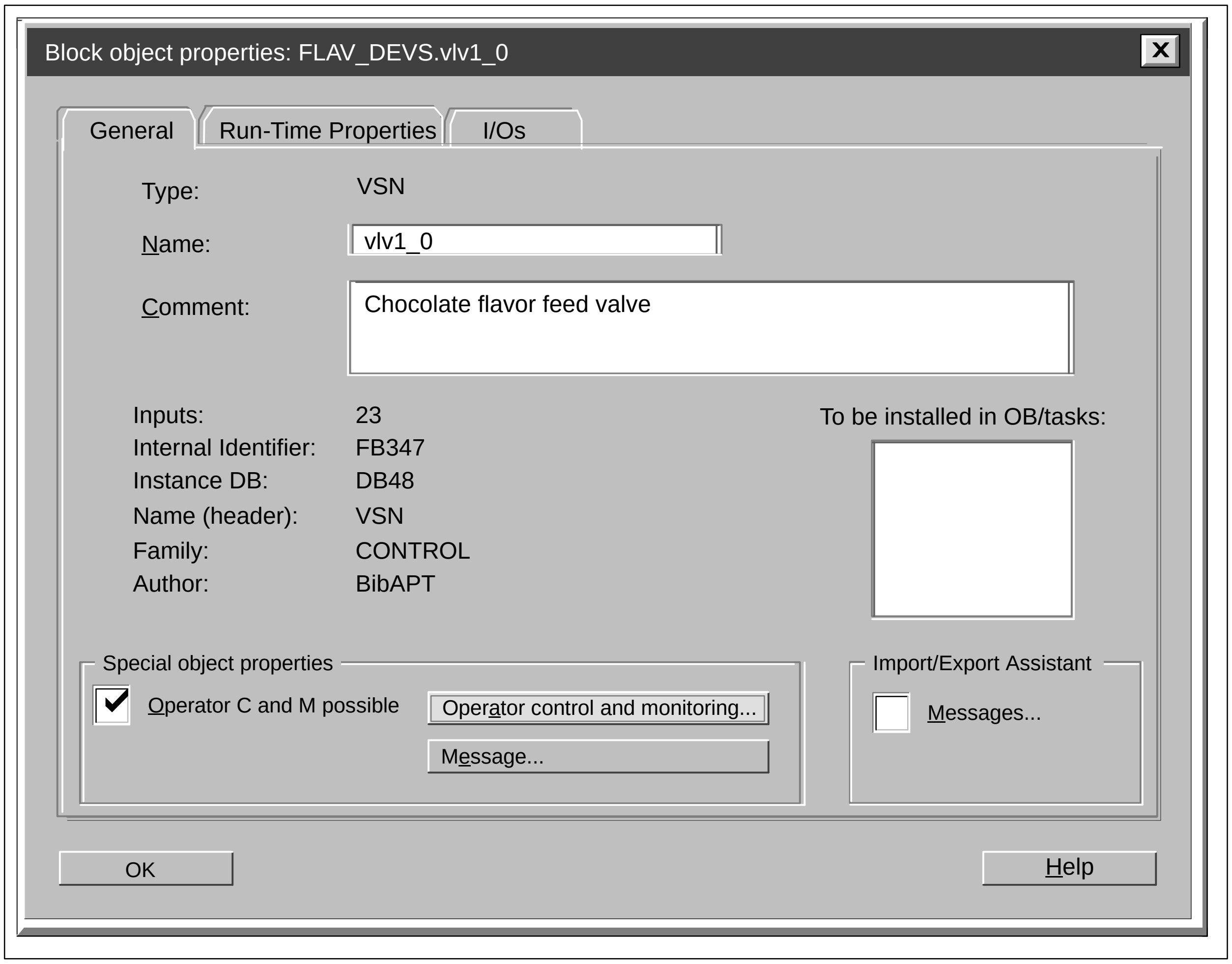
To mark the blocks and assign the tag name, follow these steps:

1. Double-click the left mouse button on the title bar of the function block. The Block Object Properties dialog box appears (Figure 2-1).
2. In the Name field, enter a unique name for the tag. The length of the tag name must be 12 characters or less and contain no spaces. The tag name can contain only letters, numbers, and underscore characters. It is recommended that you use only lower-case letters. The tag name cannot be just a number.
3. Make sure that the Operator C & M Possible field is checked. This is the default for OSx library blocks.

   No user configuration is required in the Import/Export Assistant field or for the **Message** button. Changes made from the **Message** button may result in warnings during the transfer process.
4. Click the **Operator control and monitoring...** button. The Operator Control and Monitoring window appears. The tag name appears in the Name field preceded by its path in the STEP 7 hierarchy.

Continue on Page 2-10.

This procedure continues on page 2-10.


Block object properties: FLAV_DEVS.vlv1_0
General
Run-Time Properties
I/Os

Type: VSN

Name: vlv1_0

Comment: Chocolate flavor feed valve

Inputs:	23
Internal Identifier:	FB347
Instance DB:	DB48
Name (header):	VSN
Family:	CONTROL
Author:	BibAPT

To be installed in OB/tasks:

Special object properties

✓ Operator C and M possible

Operator control and monitoring...
Message...
Import/Export Assistant

Messages...

Figure 2-1 Block Object Properties

5. You can enter a tag description for OSx purposes in the Comment field of the Operator Control and Monitoring window (Figure 2-2). The tag description in the Comment field of the General folder of Block Object Properties (Figure 2-1) is for the CFC only and does not appear in OSx.

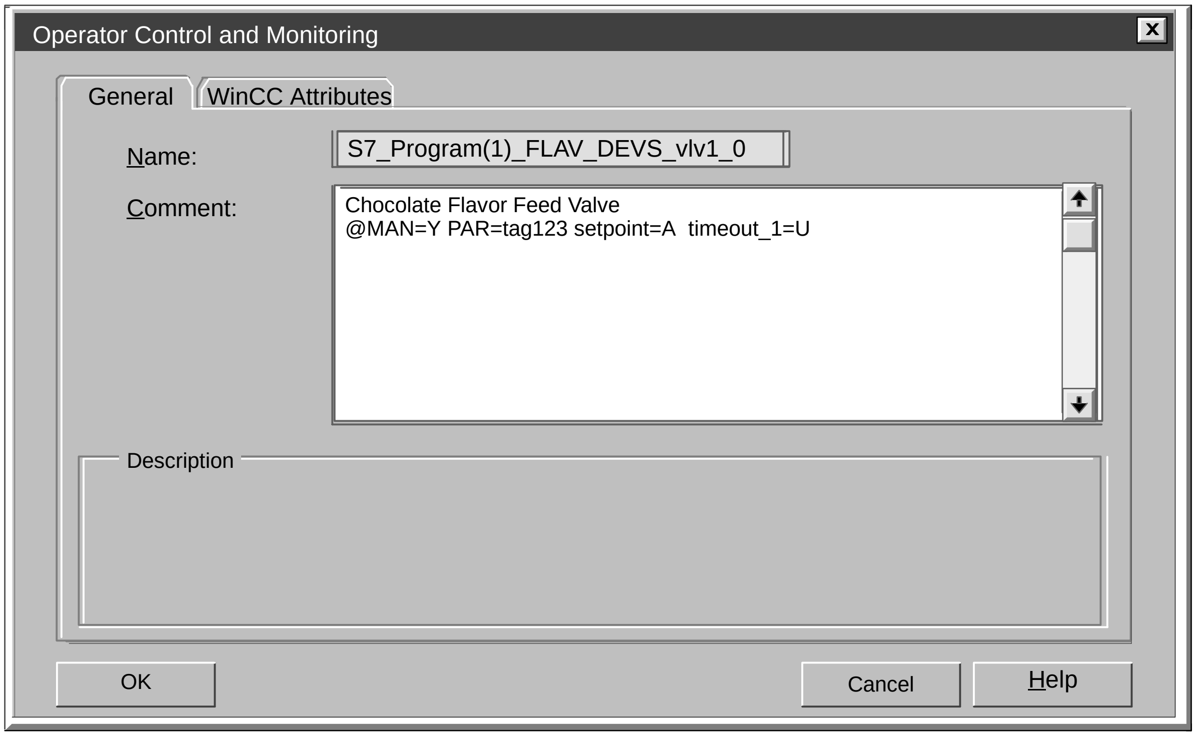
   You can also enter one or more of the following tag parameters, separated by spaces: parent unit, manual set, engineering units, autolog, and upload. See page 2-11 for descriptions of these parameters.

   No spaces are allowed within each tag parameter assignment. If the string you are assigning to a text tag attribute contains spaces, enclose the string in single quotation marks; for example:

   ```
   text_1='Unit operation complete.'
   ```

   Multiple lines are possible. You can also enter a comment here for documenting your program.

6. Click **OK** to save. Click **OK** on the Block Object Properties dialog box to exit.



S7_Program(1)_FLAV_DEVS_vlv1_0

Chocolate Flavor Feed Valve
@MAN=Y PAR=tag123 setpoint=A timeout_1=U

Figure 2-2 Operator Control and Monitoring Window

Configuring Tag Parameters in the Comment Field

The OSx tag parameters that you can enter in the Comment field of the Operator Control and Monitoring window are described below. You can defer this configuration until the tags are installed on the OSx. Once the tags are installed, you can select a tag in the Tag Configurator and make the appropriate assignments. Keep in mind, however, that any changes you make after the transfer will be overwritten the next time the file is transferred from STEP 7.

NOTE: The content of this Comment field is transferred to OSx when the function block is mapped to OSx. Changes to the Comment field do not take effect until the next mapping.

The Comment field parameters are case-sensitive and must be entered in uppercase letters; for example, `MAN=Y`.

To enter a double-quote mark (”) in the Comment field, you must enter two of them (that is, ””) in any place where a single double-quote mark is to appear in the OSx database. Failure to do so will cause errors when the tag is processed for installation on OSx.

Tag description All characters from the beginning of the field to a maximum of 30 characters or to the first occurrence of an “at” sign (@) are translated as the tag description. This means that the @ sign cannot be used as part of the description itself.

If you do not include a description, you must still enter the @ sign if you want to enter any other parameters for the tag.

Tag name change `TGN=`<OSx tag name>, where <OSx tag name> is the name to be used for the tag in OSx. You configure this parameter only if you want the tag name in OSx to be different from the instance name in the Engineering Toolset.

Change CHANGE=<n>, where n is the new deadband value, between 0.0 and 100.0; for example, CHANGE=2 for two percent. An OSx deadband is used to specify the change (in percent of span) in the input value that is required for OSx to update the value in the database. The deadband is automatically set to a default of 1.0%. Set this deadband value to filter out noise in the input signal. This value is used only to provide a setting for the OSx system.

On a multi-station system, the values displayed for a tag attribute can differ between stations by the percentage of the change setting. You can control this display of values by setting the default value of this parameter to a lower number, such as 0.1 or 0.5. If it is important to observe every numerical value change, you can even enter 0.0 for this value. However, entering 0.0 for too many attributes can result in heavy database update and display activitiy.

Parent unit PAR=<unit tag name>, where <unit tag name> is the unit tag used as the parent unit, if you assign the tag to a unit.

Manual set MAN=<Y or N> Enter Y if you want to be able to enter data manually for the tag for trending purposes.

Engineering units U=<units>, where units can be a maximum of eight characters and are only applicable to the following tags: AI, AO, LOOP, TMR, CTR, CALC, IVAR, and TEXT. If you include a space in the engineering unit, you must enclose the entire string in single quotes; for example, U='cu ft'.

Autolog and **Upload** If you select autolog, a message is printed whenever a changed value for the attribute is sent to the database. When the status attribute is specified for autolog, it is treated as an RBE tag if the OSx function block provides either an ALARM_8P or an ALARM_S messaging.

To improve network performance, select upload for ranges and alarm limit values that rarely or never change. This causes OSx to read the tag attribute at a slower rate, every other event scan period, compared to every event scan period for attributes without upload selected. Table 2-2 lists tag attributes that have upload as a default.

Table 2-2 Attributes with Upload as the Default

Tag Type	Attributes
AI	**L_RANGE, H_RANGE, HH_ALM, H_ALM, L_ALM, LL_ALM, H_DEV, L_DEV, ROC_ALM**
LOOP	**L_RANGE, H_RANGE, HH_ALM, H_ALM, L_ALM, LL_ALM, H_DEV, L_DEV, GAIN, RATE, RESET, ROC_ALM**
AO, CALC, TMR	**L_RANGE, H_RANGE**

Autolog and upload can be specified for any number of applicable tag attributes. To set these parameters for one or more tag attributes, follow the syntax below:

- For autolog only, enter <attribute>`=A`
- For upload only, enter <attribute>`=U`
- For both autolog and upload, enter <attribute>`=AU`

Text strings (for text tags only) `text_n=`<string[30]> where n is either 1, 2, or 3 corresponding to the **TEXT_1**, **TEXT_2**, or **TEXT_3** attributes.

For example, to have text_2 initially set to BATCH WAITING TO BE STARTED, you enter `text_2='BATCH WAITING TO BE STARTED'` in the Comment field. Always enclose the text string in single quotes.

Non-networked attributes Non-networked attributes for OSx can also be set in the Comment field of the Operator Control and Monitoring window. For example, to change the **H_RANGE** and **L_RANGE** limits for an IVAR, enter `H_RANGE=<n>` or `L_RANGE=<n>`, where `n` is the new value for that limit.

Table 2-3 lists the non-networked attributes for the standard OSx tag types that can be configured in the Operator Control and Monitoring window.

Table 2-3 Non-networked Attributes

Tag	Attribute
AI	AL_DEADBAND
	CHANGE
	UNITS
AO	CHANGE
	UNITS
CALC	CHANGE
	UNITS
CTR	L_RANGE
	UNITS
IVAR	H_RANGE
	L_RANGE
	UNITS
LOOP	AL_DEADBAND
	CHANGE
	UNITS
RECIPE_AREA	SCALE_HIGH
	SCALE_LOW
TEXT	UNITS
TMR	CHANGE
	UNITS
UNIT	TYPE

Process group `GRO=`<hex value> The hex value that you enter here represents the process groups to which the tag belongs; for example, `GRO=0x0000004e`. In OSx each tag is assigned by default to all 32 process groups (0xFFFFFFFF). If you prefer to assign the tag to specific process groups, see the discussion below for determining the value. For more information about process groups in OSx, see the *SIMATIC PCS 7 OSx Process Configuration Manual*.

Calculating the Hex Number for Process Groups

The hex value that you enter in the Comment field of the Operator Control and Monitoring window represents the process groups to which the function block (or symbol) belongs. The value has the form `GRO=0x00000000`, where each digit represent four process groups (Figure 2-3).

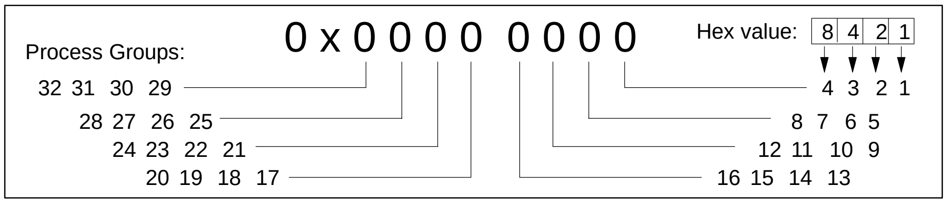


Figure 2-3 Hex Values for Process Groups

Table 2-4 shows examples of assigning individual process groups by hex number in the two left columns, and assigning multiple groups in the two right columns. The hex number for multiple process groups is obtained by adding the individual group hex numbers together. Every possible combination of the 32 process groups produces a unique hex number.

Table 2-4 Examples of Process Group Assignment

Process Group	Hex Number	Process Groups	Hex Number
1	0x00000001	1 and 2	0x00000003
2	0x00000002	1, 2, and 3	0x00000007
3	0x00000004	3 and 4	0x0000000C
4	0x00000008	1, 2, 3, and 4	0x0000000F
5	0x00000010	1, 2, 3, and 5	0x00000017
6	0x00000020	3, 4, and 6	0x0000002C

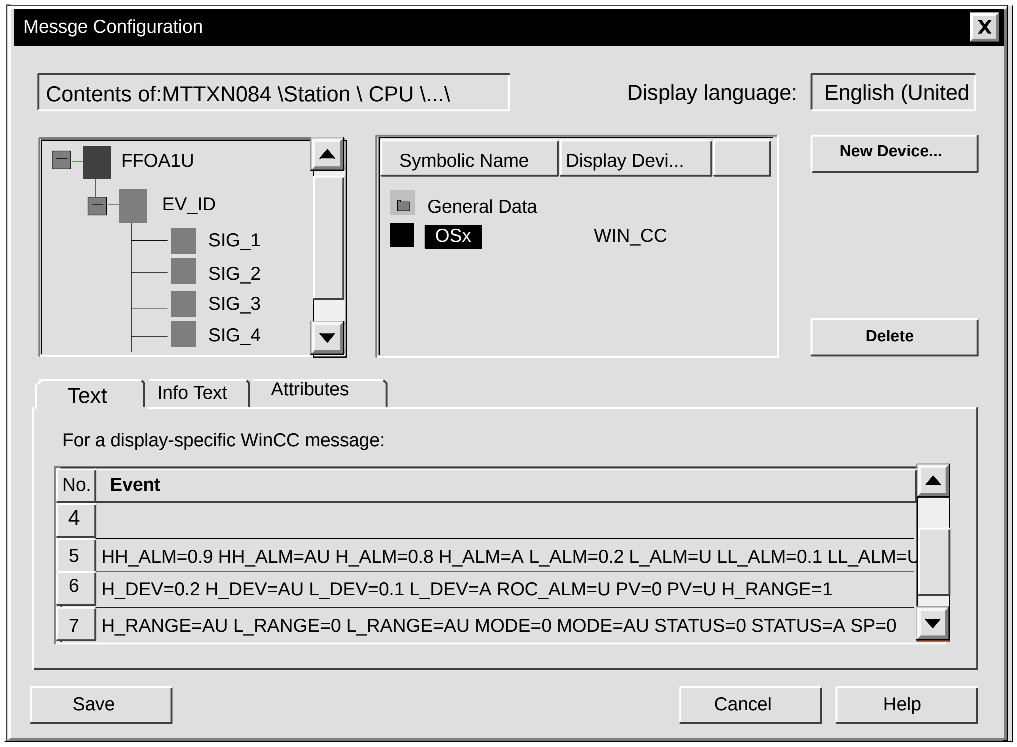
Additional Space for Tag Parameters in the Comment Field

Extra space for tag parameters is available in the text fields of the Message Configuration window (Figure 2-4). To access this window, click the **Messages...** button in the Special Object Properties field on the General tab of the Block Object Properties dialog box. You can enter any of the tag parameters described above in these fields except for the Tag Description.

This extra space is available only for function blocks that incorporate an S7 messaging block (ALARM_8P or ALARM_S). ALARM_8P blocks have eight "signal numbers" (shown as SIG_1 through SIG_8 in the figure), each of which provides a set of ten text fields. ALARM_S blocks have only one set of ten text fields. Fields 5 through 10 in each signal number are available for tag parameter entry.

Text Field 1 of each signal number that is to be transferred to OSx must contain the string $$AKZ$$ and nothing else. If this field is not initialized in this manner, the text fields for the signal number are not transferred to OSx, and the tag parameter entries in them have no effect on the OSx tag derived from the function block.

Each tag parameter entered in a message block text field must be complete within a single text field. Individual parameters must be separated by one or more spaces.


Messge Configuration

Contents of:MTTXN084 \Station \ CPU \...\

Display language: English (United

FFOA1U
EV_ID
SIG_1
SIG_2
SIG_3
SIG_4

Symbolic Name	Display Devi...	
General Data		
OSx	WIN_CC	

New Device...
Delete

Text | Info Text | Attributes

For a display-specific WinCC message:

No.	Event
4	
5	HH_ALM=0.9 HH_ALM=AU H_ALM=0.8 H_ALM=A L_ALM=0.2 L_ALM=U LL_ALM=0.1 LL_ALM=U
6	H_DEV=0.2 H_DEV=AU L_DEV=0.1 L_DEV=A ROC_ALM=U PV=0 PV=U H_RANGE=1
7	H_RANGE=AU L_RANGE=0 L_RANGE=AU MODE=0 MODE=AU STATUS=0 STATUS=A SP=0

Save | Cancel | Help

Figure 2-4 Message Configuration Dialog Box

Initial Values for Networked Attributes

You can set an initial value for inputs and outputs for an OSx function block in two separate property folders: the Block Object Properties I/O folder and the WinCC Attributes folder. The initial value in each folder has a different function. The initial value in the WinCC Attributes folder is used only with Version 4.01 of the Engineering Toolset.

When you set the initial value of an input for a networked attribute in the Block Object Properties I/O folder, that value is used for block execution and is placed in the OSx database when the system goes to the Operate state. Always provide an initial value for parameters, such as ranges and alarm limits, that are not connected in the CFC.

For example, to set an initial value of 50.0 for the setpoint of a loop, enter 50.0 in the Block Object Properties I/O folder for the **SP** input. (This example assumes that the **SP** input is not cascaded or computed.) When the tag is installed in OSx, the setpoint is initially set to zero, but when the system goes to the Operate state, the value of 50.0 is uploaded to the database.

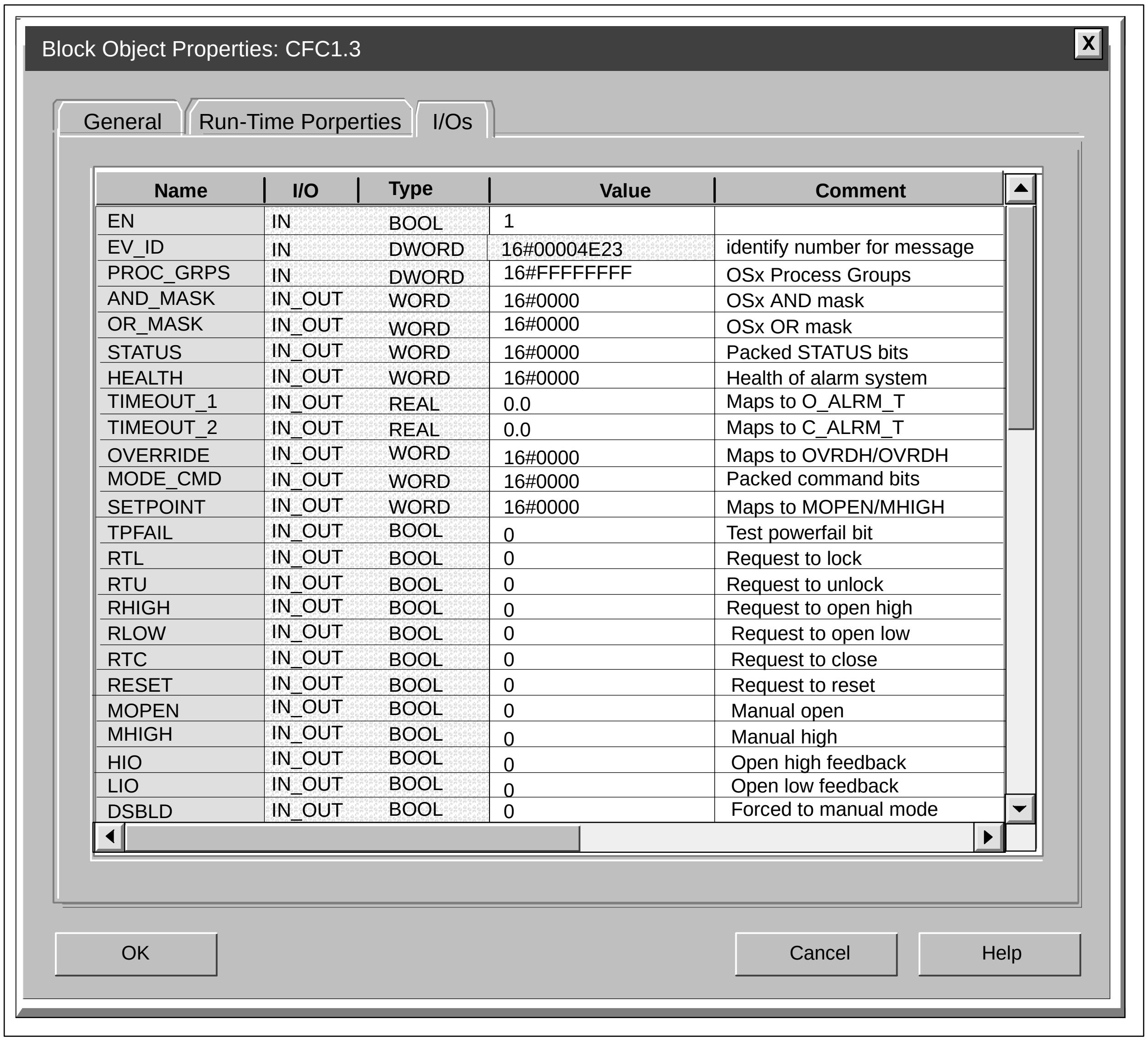
The initial value in the WinCC Attributes folder is used only for the purpose of providing an intial value when the tag is installed in OSx. If you set the initial value for a networked attribute in the WinCC Attributes folder, that value is loaded in the OSx database when the tag is installed. However, the database will be updated by the value in the Block Object Properties I/O folder when the system goes to the Operate state. Specifying a WinCC initial value can be useful either for documentation or for viewing the tag in OSx when the system is in the Offline state.

For example, you set an initial value of 50.0 in the Block Object Properties folder and an initial value of 70.0 in the WinCC Attributes folder. The tag is installed with a setpoint of 70.0 in the OSx database, but when the system goes to the Operate state, the value is updated to 50.0.

Configuring Initial Value in Block Object Properties

You can use the Value field in the Block Object Properties I/O folder to set the initial value for OSx tag attributes and for block execution. Follow the steps below:

1. Double-click the left mouse button on the title bar of the function block. The Block Object Properties dialog box appears.
2. Click the **I/Os** tab to display the list of inputs and outputs for the block (Figure 2-5).
3. Enter the values that you want use as initial values for block execution.
4. Click **OK** to confirm.



Block Object Properties: CFC1.3

Name	I/O	Type	Value	Comment
EN	IN	BOOL	1	
EV_ID	IN	DWORD	16#00004E23	identify number for message
PROC_GRPS	IN	DWORD	16#FFFFFFFF	OSx Process Groups
AND_MASK	IN_OUT	WORD	16#0000	OSx AND mask
OR_MASK	IN_OUT	WORD	16#0000	OSx OR mask
STATUS	IN_OUT	WORD	16#0000	Packed STATUS bits
HEALTH	IN_OUT	WORD	16#0000	Health of alarm system
TIMEOUT_1	IN_OUT	REAL	0.0	Maps to O_ALRM_T
TIMEOUT_2	IN_OUT	REAL	0.0	Maps to C_ALRM_T
OVERRIDE	IN_OUT	WORD	16#0000	Maps to OVRDH/OVRDH
MODE_CMD	IN_OUT	WORD	16#0000	Packed command bits
SETPOINT	IN_OUT	WORD	16#0000	Maps to MOPEN/MHIGH
TPFAIL	IN_OUT	BOOL	0	Test powerfail bit
RTL	IN_OUT	BOOL	0	Request to lock
RTU	IN_OUT	BOOL	0	Request to unlock
RHIGH	IN_OUT	BOOL	0	Request to open high
RLOW	IN_OUT	BOOL	0	Request to open low
RTC	IN_OUT	BOOL	0	Request to close
RESET	IN_OUT	BOOL	0	Request to reset
MOPEN	IN_OUT	BOOL	0	Manual open
MHIGH	IN_OUT	BOOL	0	Manual high
HIO	IN_OUT	BOOL	0	Open high feedback
LIO	IN_OUT	BOOL	0	Open low feedback
DSBLD	IN_OUT	BOOL	0	Forced to manual mode

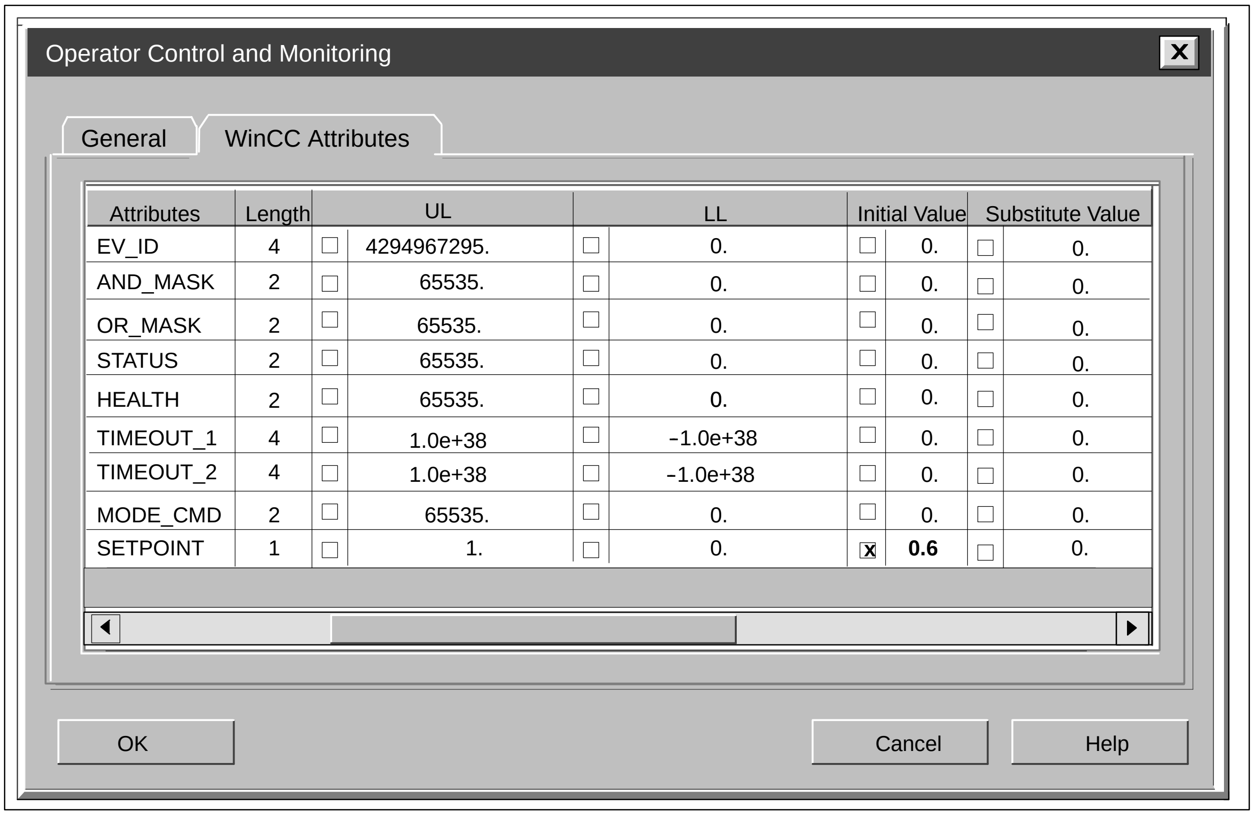
Figure 2-5 Setting Initial Value in the Block Object Properties I/O Folder

Configuring Initial Value in WinCC Attributes

If you use Version 4.01 of the Engineering Toolset, you can use the Initial Value field in the WinCC Attributes folder to set the initial value for OSx tag attributes. Remember that, for a networked attribute, this value will be overwritten by the value in the Block Object Properties I/O folder when the system goes to the Operate state.

To set the initial value in the WinCC Attributes folder, follow the steps below:

1. Double-click the left mouse button on the title bar of the function block. The Block Object Properties dialog box appears.
2. Click the **Operator Control and Monitoring** button.
3. Click on the **WinCC Attributes** tab. The WinCC Attributes folder is displayed (Figure 2-6).
4. In the Initial Value field, click the box to check it, and enter the value that you want use as the initial value in the OSx database. Remember that, for a networked attribute, this value will be overwritten by the value in the Block Object Properties I/O folder when the system goes to the Operate state.
5. Click **OK** to confirm.



Operator Control and Monitoring

Attributes	Length		UL		LL		Initial Value		Substitute Value
EV_ID	4	☐	4294967295.	☐	0.	☐	0.	☐	0.
AND_MASK	2	☐	65535.	☐	0.	☐	0.	☐	0.
OR_MASK	2	☐	65535.	☐	0.	☐	0.	☐	0.
STATUS	2	☐	65535.	☐	0.	☐	0.	☐	0.
HEALTH	2	☐	65535.	☐	0.	☐	0.	☐	0.
TIMEOUT_1	4	☐	1.0e+38	☐	–1.0e+38	☐	0.	☐	0.
TIMEOUT_2	4	☐	1.0e+38	☐	–1.0e+38	☐	0.	☐	0.
MODE_CMD	2	☐	65535.	☐	0.	☐	0.	☐	0.
SETPOINT	1	☐	1.	☐	0.	☒	**0.6**	☐	0.

Figure 2-6 Setting Initial Value in the WinCC Attributes Folder

2.3 Creating Tags from Symbols

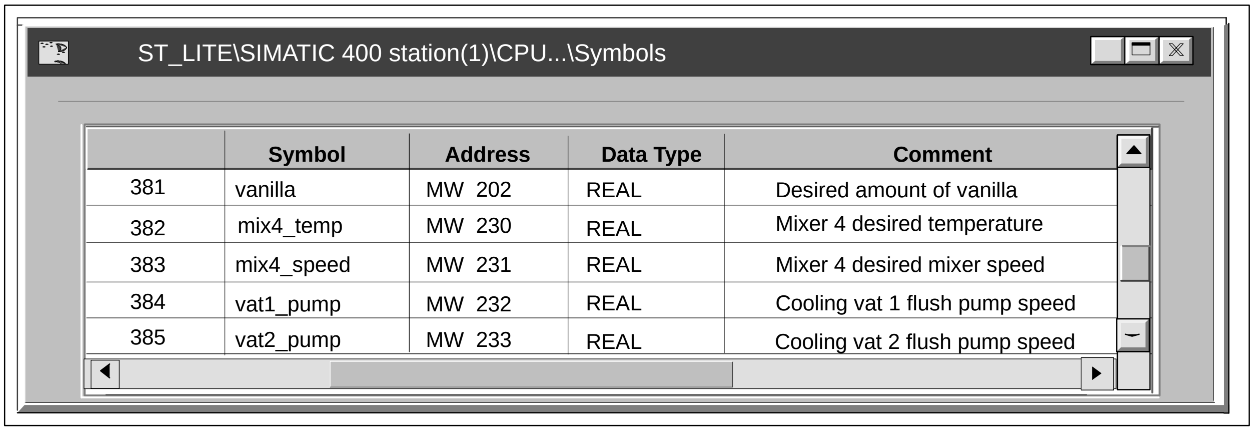
Assigning Symbol Names

A symbol is a name that you assign in the symbol table to the flag memory area that can then be used throughout the S7 program, and can also be translated as a tag attribute to OSx. Refer to the *SIMATIC STEP 7 User Manual* for more information on using the Symbol Table Editor.

NOTE: It is recommended that you use the OSx Library blocks for most tag types since the blocks are preconfigured for OSx compatibility. If you use symbols for tag types, you must specify one symbol for each attribute.

NOTE: Function blocks require more memory in the S7 controller than symbols. To reduce memory requirements on the controller, you can use symbols in place of function blocks for tags that do not require alarm messaging. For example, if a limit switch does not require alarming, then you can use a symbol instead of the DI function block.

To access the symbol table, highlight the S7 program in the SIMATIC Manager and double-click on the **Symbols** icon. Enter the symbol name in the Symbol column of the table. Figure 2-7 shows a section of a symbol table for an S7 program.



	Symbol	Address	Data Type	Comment
381	vanilla	MW 202	REAL	Desired amount of vanilla
382	mix4_temp	MW 230	REAL	Mixer 4 desired temperature
383	mix4_speed	MW 231	REAL	Mixer 4 desired mixer speed
384	vat1_pump	MW 232	REAL	Cooling vat 1 flush pump speed
385	vat2_pump	MW 233	REAL	Cooling vat 2 flush pump speed

Figure 2-7 Symbol Table

The name of a symbol that you want to translate as a tag to OSx consists of two parts: a tag name and a suffix.

- The tag name must be 12 characters or less (not counting the suffix) and contain no spaces. The tag name can contain only letters, numbers, and underscore characters. It is recommended that you use only lower-case letters for the tag name.
- The suffix is an upper-case two-character code identifying the OSx tag attribute to which the symbol is to be mapped. See Table 2-5 on page 2-24 for a listing of these codes.

The general format for a symbol is as follows:

```
tag_nameXY
```

where `X` encodes the tag and `Y` encodes the attribute. The `XY` suffix is removed when the tag is installed in the OSx database.

For any OSx tag type, only the symbols that are useful to the program need to be configured in the symbol table. For example, an IVAR:STATUS is not required when an IVAR:VALUE is configured. Similarly, if only TEXT: TEXT_1 is useful, then you do not need to configure TEXT:TEXT_2 and TEXT:TEXT_3.

Table 2-5 shows the tag attributes supported for symbols to be used as tags in OSx, valid memory areas, data format, and data size using the two-character code. The column on the far right shows examples of the symbols.

Table 2-5 Tag Attributes for Symbols

Tag:Attribute	Address [1]	Data Type	Size	Tag Code	Attribute Code	Example
CALC:STATUS	M	BOOL	1 bit	C	S	tag_oneCS
CALC:VALUE	MD	REAL	2 words	C	V	tag_oneCV
CALC:CHANGE	MD	REAL	N/A [2]	C	C	tag_oneCC
IVAR:STATUS	M	BOOL	1 bit	I	S	tag_twoIS
IVAR:VALUE	MW	INT	1 word	I	V	tag_twoIV
TEXT:STATUS	M	BOOL	1 bit	T	S	tag_threeTS
TEXT:TEXT_1	MW	WORD	15 words [3]	T	1	tag_threeT1
TEXT:TEXT_2	MW	WORD	15 words [3]	T	2	tag_threeT2
TEXT:TEXT_3	MW	WORD	15 words [3]	T	3	tag_threeT3
DI:STATUS	M or MW	BOOL or WORD	1 bit or word	X	S	tag_fourXS
DO:STATUS	M	BOOL	1 bit	Y	S	tag_fiveYS
DO:COMMAND	M	BOOL	1 bit	Y	C	tag_fiveYC
AO:STATUS	M	BOOL	1 bit	A	S	tag_sixAS
AO:CHANGE	MD	REAL	N/A [2]	A	C	tag_sixAC
AO:OUT	MD	REAL	2 words	A	O	tag_sixAO
AO:MODE	M	BOOL	1 bit	A	M	tag_sixAM
DI10:STATUS	MW	WORD	1 word	W	S	tag_sevenWS
DO10:STATUS	MW	WORD	1 word	Z	S	tag_eightZS
DO10:DATA_VAL1	M	BOOL	1 bit	Z	1	tag_eightZ1
DO10:DATA_VAL2	M	BOOL	1 bit	Z	2	tag_eightZ2
DO10:DATA_VAL3	M	BOOL	1 bit	Z	3	tag_eightZ3
DO10:DATA_VAL4	M	BOOL	1 bit	Z	4	tag_eightZ4
DO10:DATA_VAL5	M	BOOL	1 bit	Z	5	tag_eightZ5
DO10:DATA_VAL6	M	BOOL	1 bit	Z	6	tag_eightZ6
DO10:DATA_VAL7	M	BOOL	1 bit	Z	7	tag_eightZ7
DO10:DATA_VAL8	M	BOOL	1 bit	Z	8	tag_eightZ8
DO10:DATA_VAL9	M	BOOL	1 bit	Z	9	tag_eightZ9
DO10:DATA_VAL10	M	BOOL	1 bit	Z	A	tag_eightZA

1 Symbols that you configure as OSx tags must be assigned only flag memory addresses, such as M, MW, or MD.
2 N/A means the OSx does not read the value from the memory address (that is, it is not networked). Instead it uses the Initial Value that you configured in the WinCC Attributes folder. The symbol requires an address in order to be mapped.
3 When the size exceeds the size of the data type, be sure to reserve the correct number of flag words for the data. For example, a text:text_1 requires 15 flag words beginning at the specified address.

Guidelines for Marking Symbols

When you mark symbols for tag translation to OSx, take the following considerations into account:

- If you do not define the **STATUS** attribute, it is automatically defined for you as a non-networked attribute.
- If you mark a symbol as a CALC or IVAR, the defaults in the OSx database will be used for the low and high ranges (0.0 to 100.0 for CALC, 0 to 100 for IVAR). If the value is likely to exceed these ranges, you must specify new values for the high and/or low ranges, as described in the procedure on page 2-26.

NOTE: When you configure AO, CALC, or IVAR tags as symbols using the Engineering Toolset, the default values for high_range and low_range are not valid for the OSx database. These default values cause an error on the OSx station when you try to install these tags, and no tags are installed.

When you configure AO, CALC, and IVAR tags as symbols, set the upper and lower limits according to the procedure on page 2-26. Enter the limit values that are to be used on the OSx station.

- The **CHANGE** attribute (database deadband) has a range of 0.0 to 100.0 and is set at a default value of 1.0, which means that the value in the OSx database is updated whenever the value changes by 1% of span. For example, if you want to change the deadband for a calc_10CV, then you must also create a symbol calc_10CC and give it a smaller initial value for faster updates (0.0 updates each value change), or a larger initial value for less frequent updates.

For more information on tags and tag attributes, see the *SIMATIC PCS 7 OSx Process Configuration Manual*.

Marking Symbols

To mark a symbol for translation to OSx, follow the steps below:

1. Make sure that the symbol name is suffixed with a tag and attribute code combination from Table 2-5 on page 2-24; for example, ao_31AO. Verify that the Address and Data Type fields are filled in before you proceed.

2. Click on the symbol name in the symbol table to highlight it. Then right-click on the highlighted symbol name. A pop-up menu appears.

3. Select **Special Object Properties->Operator Control and Monitoring** on the pop-up menu. (The **Communication** and **Message** options on this menu do not apply to OSx tags.) The Operator Control and Monitoring window appears (Figure 2-8).

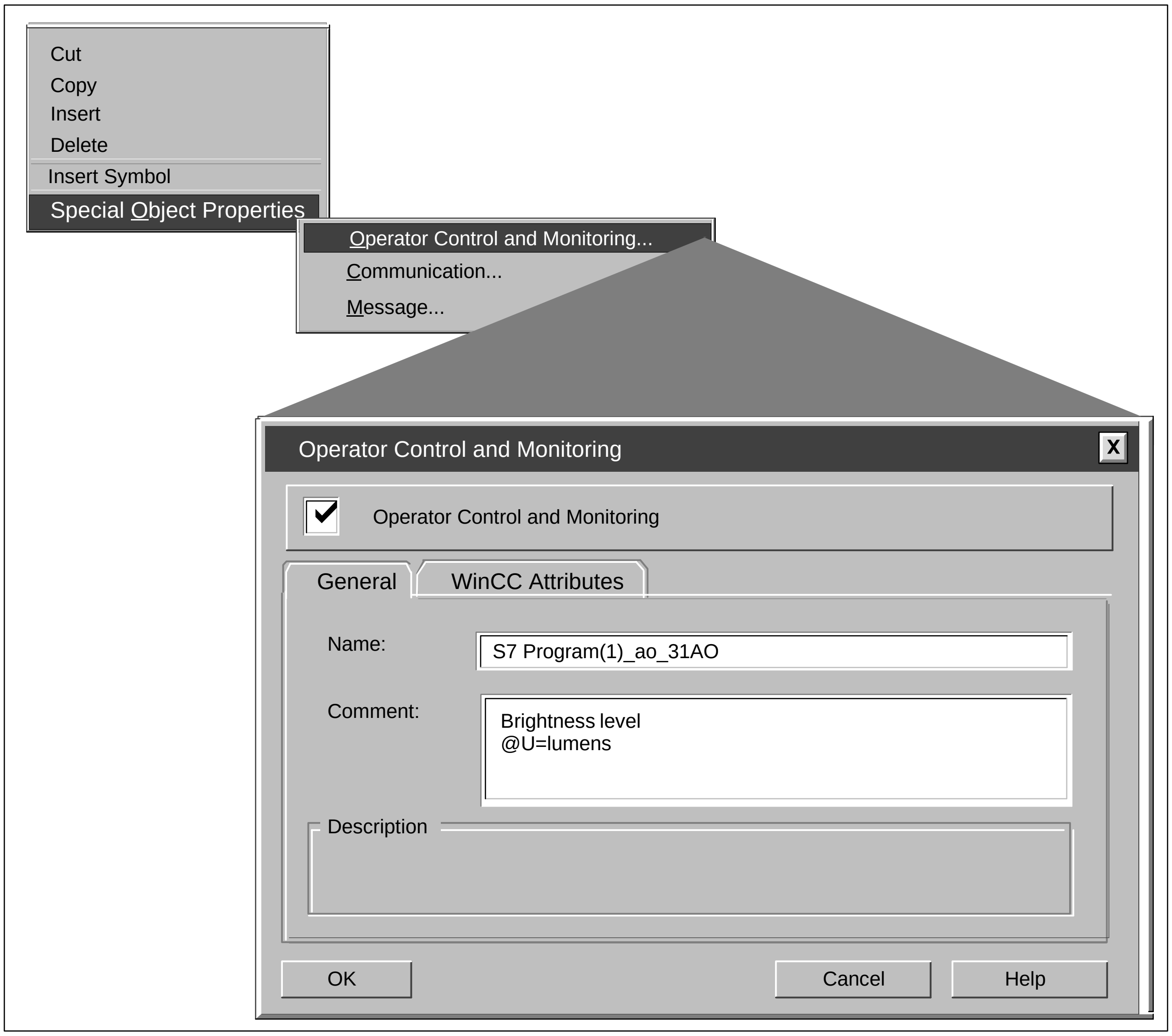
4. Check the Operator Control and Monitoring box in the upper left-hand corner of the window.

5. You can enter a tag description in the Comment field of this window. You can also enter one or more of the following tag parameters, separated by spaces: parent unit, manual set, engineering units, process group, autolog, and upload. For text tags only, you can also enter text strings. See page 2-28 for descriptions of these parameters.

6. Click the **WinCC Attributes** tab. The WinCC Attributes window appears. See Figure 2-9 on page 2-29.

7. Scroll to the right to locate the Initial Value field. Click the box in the fields to mark it. Enter a value for the field. For the tag attributes **CALC:VALUE**, **IVAR:VALUE**, and **AO:OUT**, you can also mark UL (upper limit) and LL (lower limit) fields and enter values for them. In OSx, these settings will be used for the H_RANGE and L_RANGE, respectively.

8. Click **OK** to confirm your entries.


Cut
Copy
Insert
Delete
Insert Symbol
Special Object Properties
Operator Control and Monitoring...
Communication...
Message...
Operator Control and Monitoring
X
Operator Control and Monitoring
General
WinCC Attributes

Name: S7 Program(1)_ao_31AO

Comment: Brightness level
@U=lumens

Description
OK
Cancel
Help

Figure 2-8 Setting Symbol Parameters in the Comment Field

Creating Tags from Symbols (continued)

Configuring Tag Parameters in the Comment Field

The tag parameters that you can enter in the Comment field of the Operator Control and Monitoring window are described below. You can defer this configuration until the tags are installed on the OSx. Once the tags are installed, you can select a tag in the Tag Configurator and make the appropriate assignments. Keep in mind, however, that any changes you make after the transfer will be overwritten the next time the file is transferred from STEP 7.

To enter a double-quote mark (”) in the Comment field, you must enter two of them (that is, ””) in any place where a single double-quote mark is to appear in the OSx database. Failure to do so will cause errors when the tag is processed for installation on OSx.

Tag description All characters from the beginning of the field to a maximum of 30 characters or to the first occurrence of an “at” sign (@) are translated as the tag description. This means that the @ sign cannot be used as part of the description itself.

If you do not include a description, you must still enter the @ sign if you want to enter any other parameters for the tag.

Parent unit `PAR=`<unit tag name>

Manual set `MAN=`<`Y` or `N`>

Engineering units `U=`<units>, where units can be a maximum of eight characters and are only applicable to the **UNITS** attribute of AO, CALC, IVAR, and TEXT symbols. If you include a space in the engineering unit, you must enclose the entire string in single quotes; for example, U=’cu ft’.

Autolog and **Upload** To set these parameters for the attributes of a tag follow the syntax below. For further information, see page 2-13.

- For autolog only, enter <attr_name>`=A`
- For upload only, enter <attr_name>`=U`
- For both autolog and upload, enter <attr_name>`=AU`

Text strings (for text tags only) text_`n=`<string[30]> where n is either 1, 2, or 3 corresponding to the **TEXT_1**, **TEXT_2**, or **TEXT_3** attributes.

For example, to have text_2 initially set to BATCH WAITING TO BE STARTED, you enter `text_2=’BATCH WAITING TO BE STARTED’` in the Comment field. Always enclose the text string in single quotes.

Process group `GRO=<hex value>` The hex value that you enter here represents the process groups to which the symbol belongs; for example, `GRO=0x0000004e.` See the discussion on page 2-15 for help in determining this value.

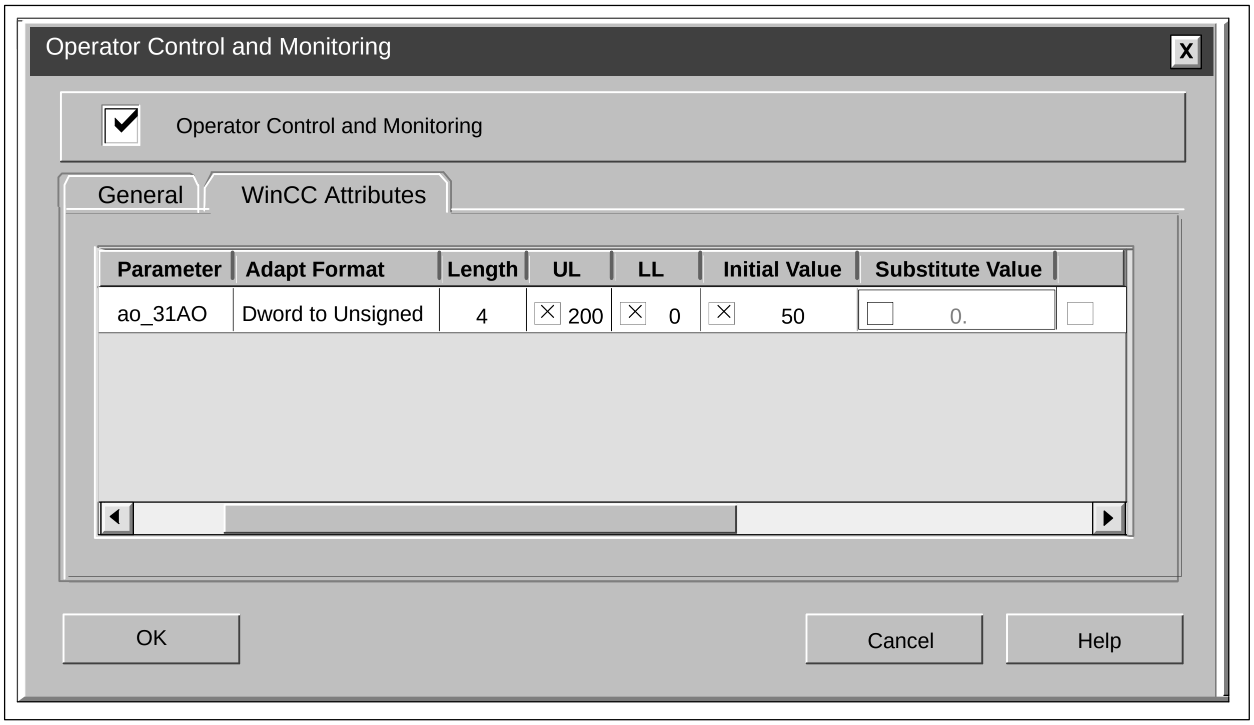
Configuring Tag Parameters in WinCC Attributes

You configure the initial value, high range, and low range of a tag in the WinCC Attributes folder (Figure 2-9).

Initial Value Use the Intial Value field on the **WinCC Attributes** tab of the Operator Control and Monitoring dialog box. Remember that, for a networked attribute, this value will be overwritten by the value in the controller when the system goes to the Operate state.

High Range (for AO, CALC, and IVAR tags only) Use the UL (upper limit) field on the **WinCC Attributes** tab of the Operator Control and Monitoring dialog box.

Low Range (for AO, CALC, and IVAR tags only) Use the LL (lower limit) field on the **WinCC Attributes** tab of the Operator Control and Monitoring dialog box.


Operator Control and Monitoring
X
Operator Control and Monitoring
General
WinCC Attributes
OK
Cancel
Help

☑ Operator Control and Monitoring

Parameter	Adapt Format	Length	UL	LL	Initial Value	Substitute Value	
ao_31AO	Dword to Unsigned	4	☒ 200	☒ 0	☒ 50	☐ 0.	☐

Figure 2-9 Assigning Initial Value, High and Low Ranges in the WinCC Attributes Folder

2.4 Translating Function Blocks and Symbols to OSx Tags: Overview

You have now created a project in STEP 7 with the Engineering Toolset, using blocks from the OSx Library and creating symbols in the Symbol Table for any tag attributes that you want to use in OSx.

You have marked the blocks and symbols and now want to install them as tags in the OSx database so that you can use them for monitoring and control of your process (in graphics, reports, recipes). To translate the marked function blocks and symbols into OSx tags, you must perform the following tasks:

- Establish a connection between the S7 project and the OSx system (Section 2.5)
- Create a mapper file for the marked blocks and symbols (Section 2.6)
- Transfer the mapper file to a disk or to an OSx station (Section 2.7)
- Install the tags from the mapper file in the OSx database (Section 2.8)

2.5 Inserting an OSx Connection

Inserting an OSx Connection

To establish the connection between the S7 project and the OSx, follow the steps below. These steps should be performed one time only per project. If the project already has the OSx icon, skip these steps.

1. Be sure that you have installed the necessary software on OSx. See Chapter 1.
2. In the SIMATIC Manager, open the project that contains the function blocks that you want to use as tags in OSx.
3. Highlight the name of the S7 project by clicking on it.
4. From the SIMATIC Manager menu bar, select **Insert–>OSx**. An OSx station is added to the project. A labeled icon appears in the project level directory (Figure 2-10).

Inserting an OSx Connection (continued)

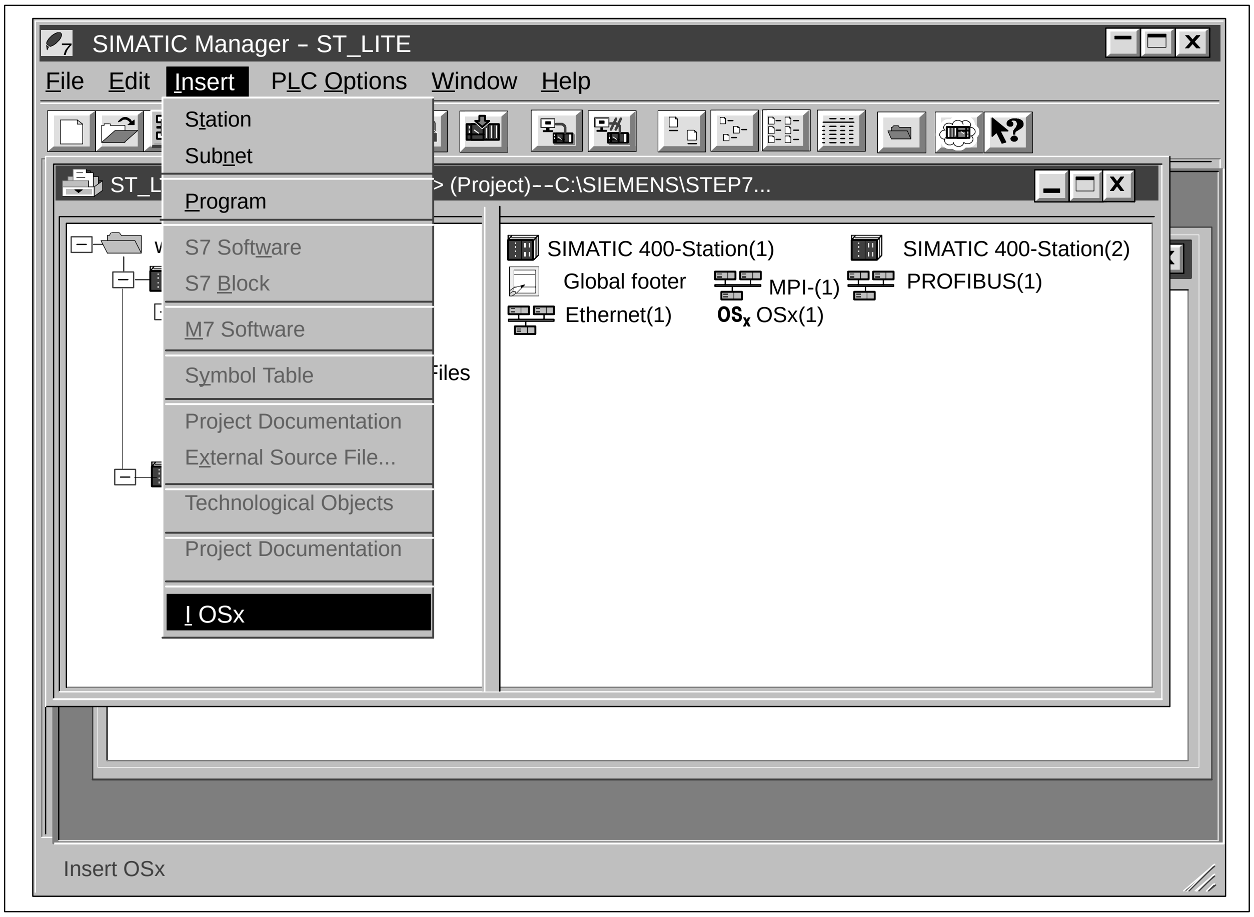
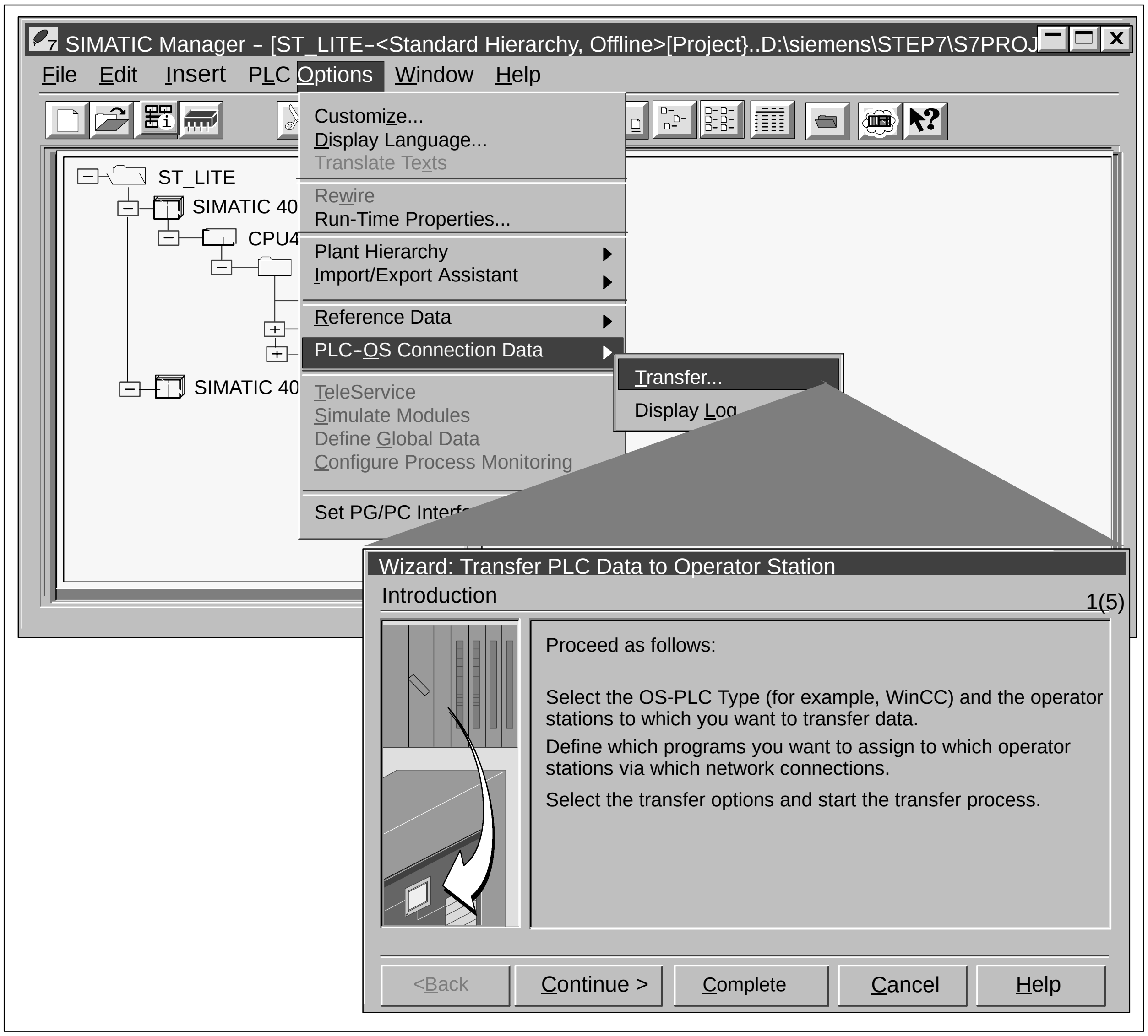


Figure 2-10 Inserting an OSx Connection

2.6 Creating the Mapper File

Before you create a mapper file for the marked blocks and symbols, confirm that the S7 program name is identical to the control node name as configured on the OSx. To create the mapper file follow the steps below:

1. From the SIMATIC Manager menu bar, select **Options-> PLC-OS Connection Data->Transfer**. The Transfer PLC Data to Operator Station wizard appears (Figure 2-11).



Proceed as follows:

Select the OS-PLC Type (for example, WinCC) and the operator stations to which you want to transfer data.

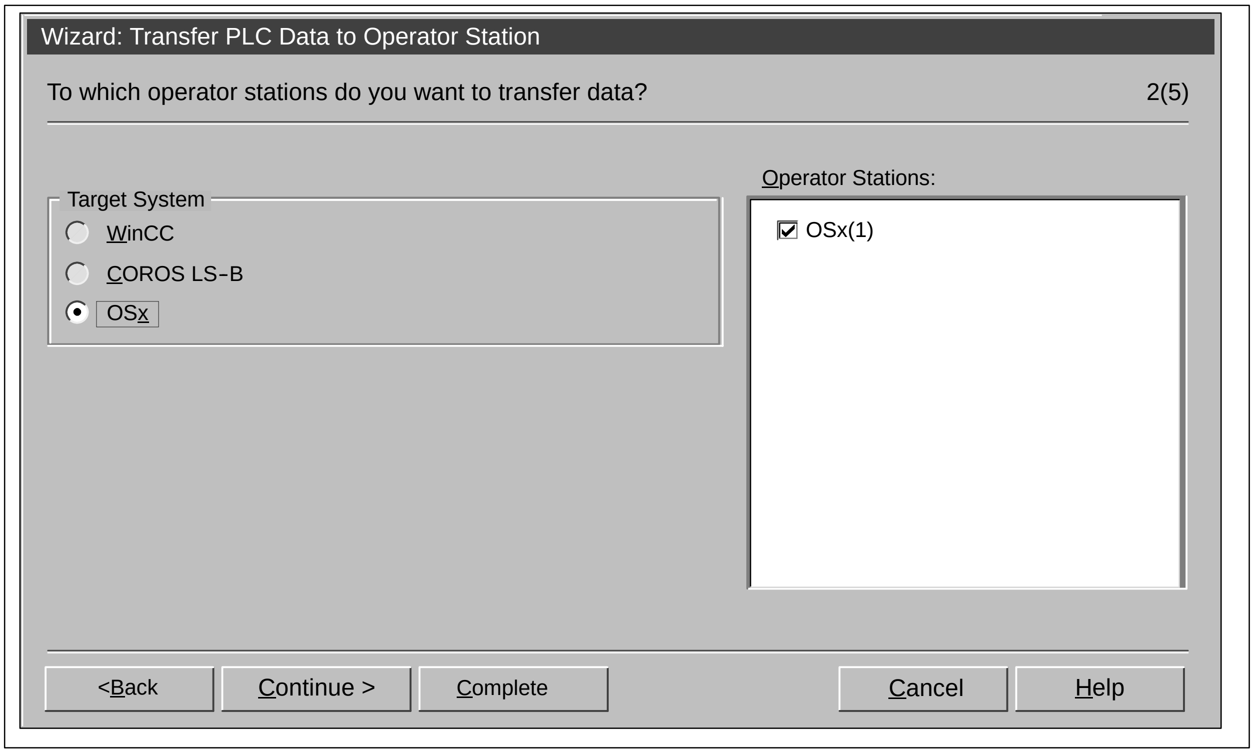
Define which programs you want to assign to which operator stations via which network connections.

Select the transfer options and start the transfer process.

Figure 2-11 Transfer PLC Data to Operator Station

2. Click **Continue**. The OS selection window appears (Figure 2-12).

3. Make sure that the OSx station in the Operator Stations field is checked.


Wizard: Transfer PLC Data to Operator Station

To which operator stations do you want to transfer data? 2(5)

Target System
- WinCC
- COROS LS-B
- [✓ OSx]

Operator Stations:
- [✓] OSx(1)

Figure 2-12 Selecting the OSx Station

4. Click **Continue**. A window appears that allows you to assign an S7 program to the OSx (Figure 2-13). If the programs have already been assigned to OSx, go to step 6.

5. In the S7/M7 Programs field, click to highlight the S7 program name that you want to assign to the OSx.

6. In the Operator Stations field, click to highlight the OSx station to which you are assigning the S7 program.

7. Click the right arrow button in the center of the screen. The program is assigned to the OSx. You can assign one or more programs to the OSx station.

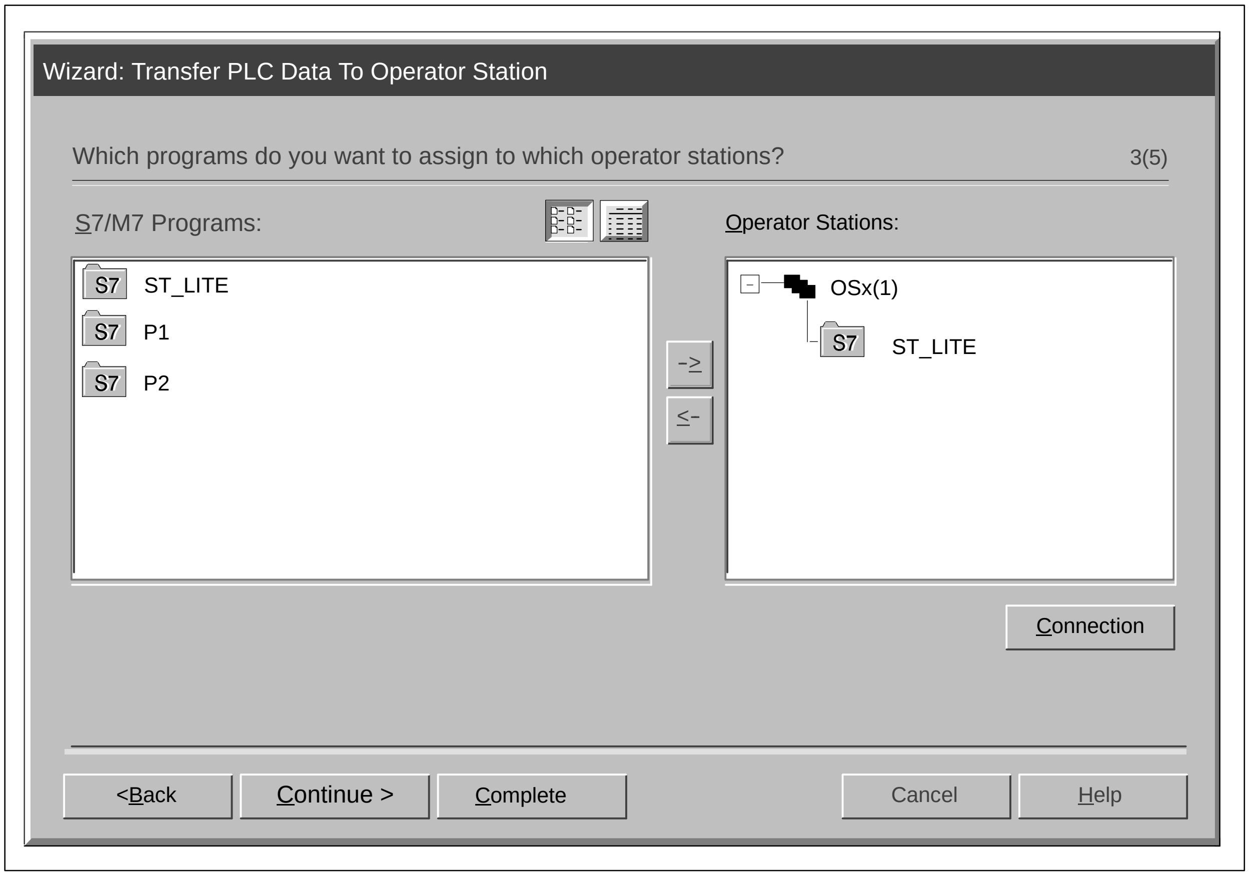
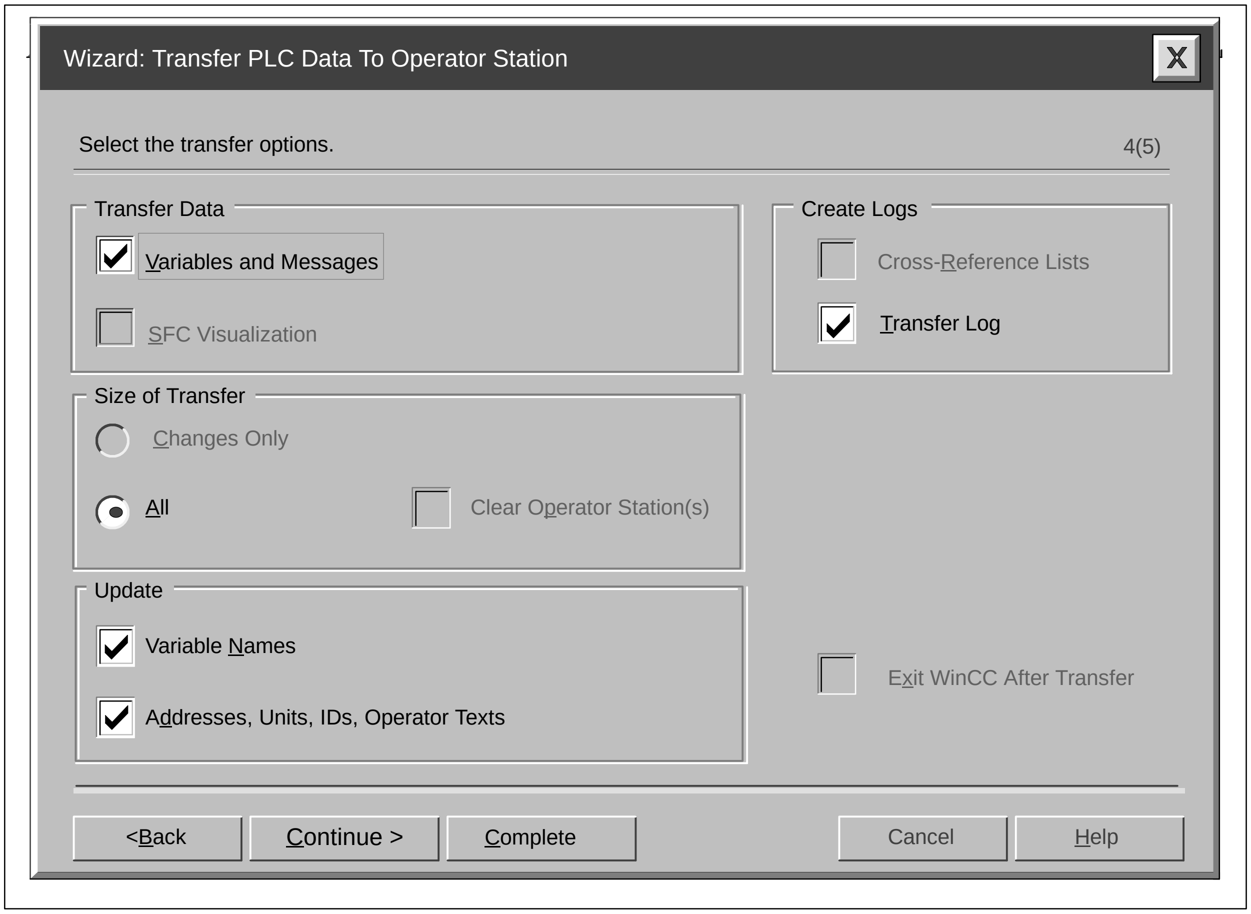


Figure 2-13 Assigning Programs to the OSx

This procedure continues on the next page.

8. Click **Continue**. A transfer options window appears. Make sure the options are checked as shown in Figure 2-14.



Wizard: Transfer PLC Data To Operator Station

Select the transfer options. 4(5)

Transfer Data
- [✓] Variables and Messages
- [ ] SFC Visualization

Create Logs
- [ ] Cross-Reference Lists
- [✓] Transfer Log

Size of Transfer
- ( ) Changes Only
- (●) All
- [ ] Clear Operator Station(s)

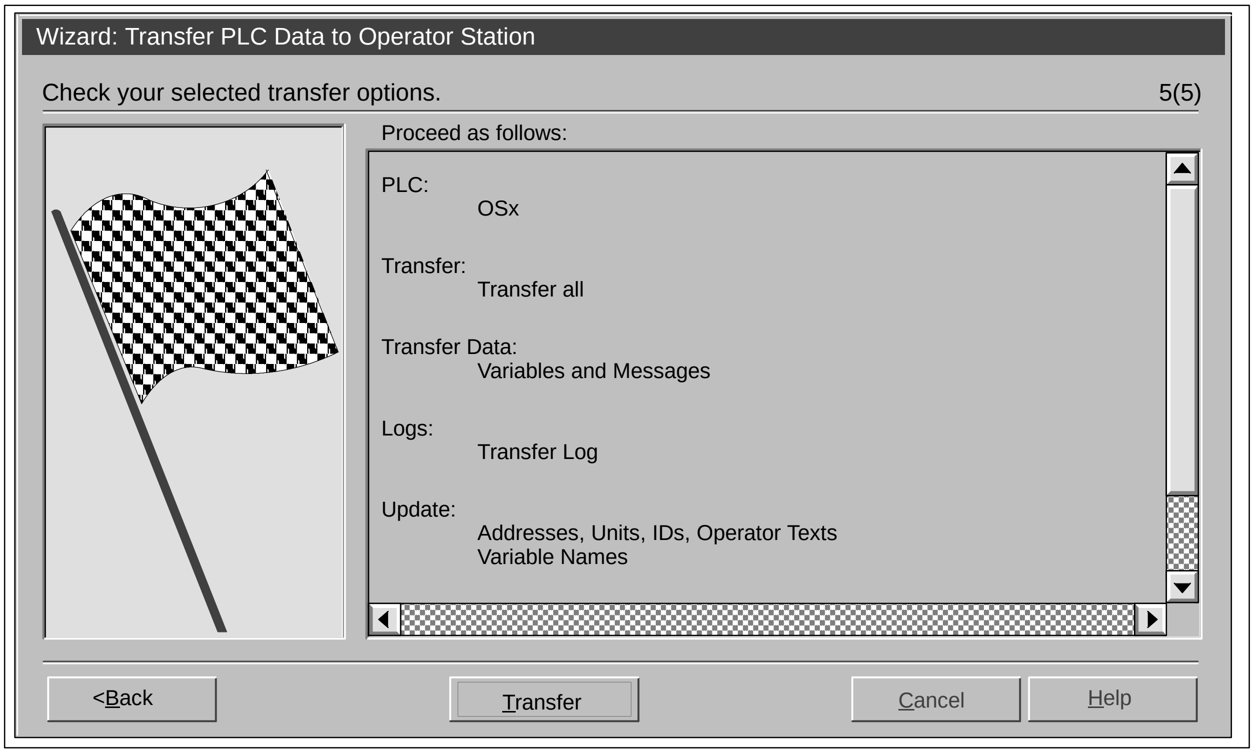
Update
- [✓] Variable Names
- [✓] Addresses, Units, IDs, Operator Texts

- [ ] Exit WinCC After Transfer

Figure 2-14 Selecting Transfer Options

9. Click **Continue**. A transfer summary window appears (Figure 2-15). Check your selections. Use the **Back** button to make any necessary corrections.

10. When you are certain that your selections are correct, click **Transfer**. Respond to the warning that appears. The transfer may take several minutes. Various windows show you the progress of the transfer.



Wizard: Transfer PLC Data to Operator Station
Check your selected transfer options.
5(5)
Proceed as follows:
<Back
Transfer
Cancel
Help

PLC:
OSx

Transfer:
Transfer all

Transfer Data:
Variables and Messages

Logs:
Transfer Log

Update:
Addresses, Units, IDs, Operator Texts
Variable Names

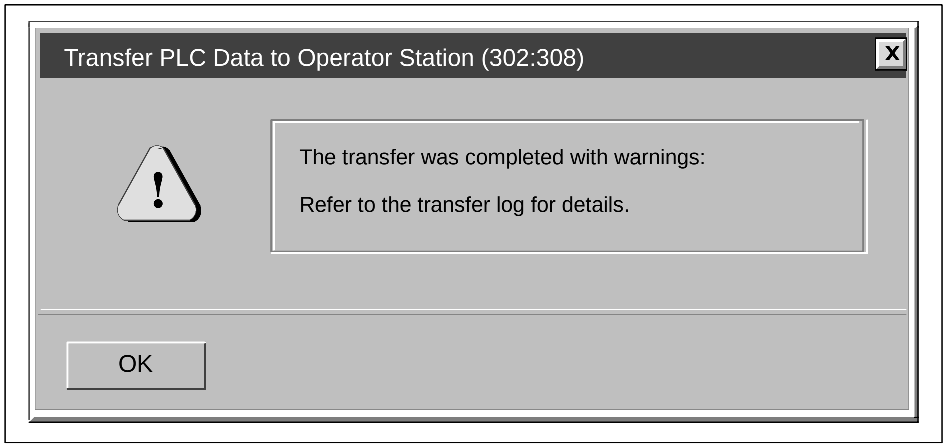
Figure 2-15 Transfer Summary Window

This procedure continues on the next page.

Creating the Mapper File (continued)

11. A message appears to tell you when the transfer is complete.

    If warnings occurred during the transfer, the message in Figure 2-16 appears. To view the warnings, select **Options->PLC-OS Connection Data->Display Log**, then use the Search or Find option to locate instances of "warning" in the **transfer.log** file. Make necessary corrections and attempt the transfer again.

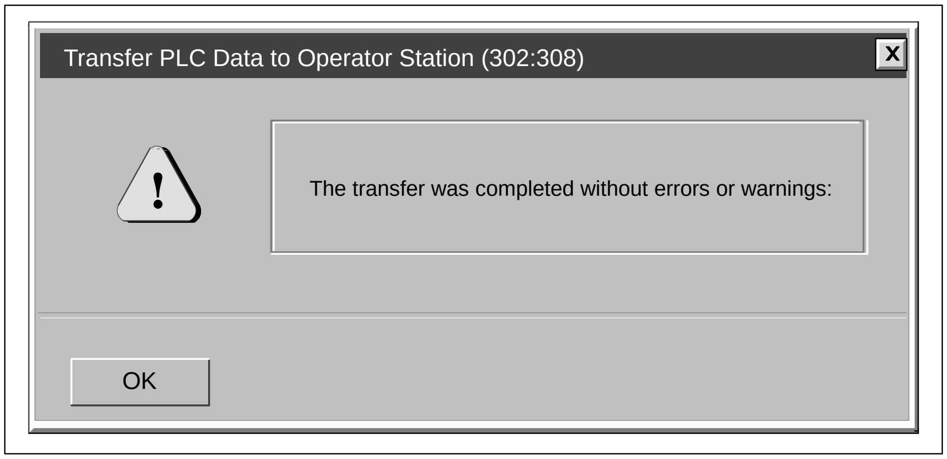


The transfer was completed with warnings:

Refer to the transfer log for details.

Figure 2-16 Transfer with Errors or Warnings

If the transfer has successfully completed without warnings or errors, the message in Figure 2-17 appears. The marked blocks and symbols have now been placed in the mapper files. Click **OK**.



The transfer was completed without errors or warnings:

Figure 2-17 Successful Transfer

2.7 Transferring the Mapper File to OSx

Once you have created the mapper file for an S7 project, you need to transfer the project to a disk or to an OSx station. Follow these steps:

1. Start the transfer program from the **Start** button at the bottom left of the Windows main screen. Select **Simatic->Transfer PLC Data to OSx**. The Transfer dialog box appears (Figure 2-18).

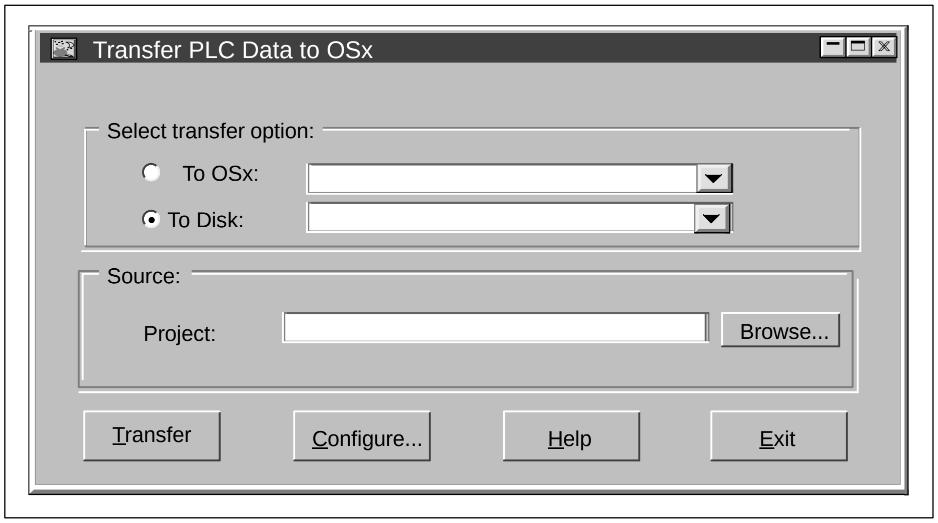


Figure 2-18 Transfer Dialog Box

This procedure continues on the next page.

2. Click on the **Browse** button. The Browse dialog box appears (Figure 2-19).

3. To create a path for the project mapper file, select the disk drive where the Engineering Toolset is installed. Double-click on the appropriate directories to take you to your S7 project (**St_lite** in the example below).

   If you plan to use this particular path on a regular basis, you can set it as the default path. See Setting a Default Path on page 2-42.

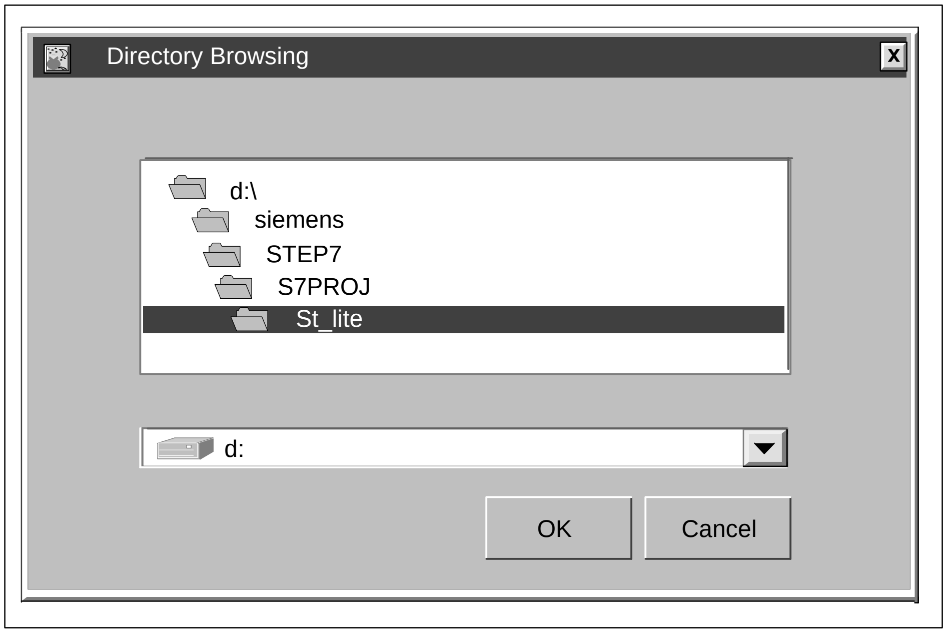


Figure 2-19 Transfer Utility: Browse

4. Click **OK**. The path appears in the Transfer dialog box (Figure 2-20).

5. At the top of the Transfer dialog box, select the destination.

   To transfer to disk, click in the circle to the left of **To Disk**.

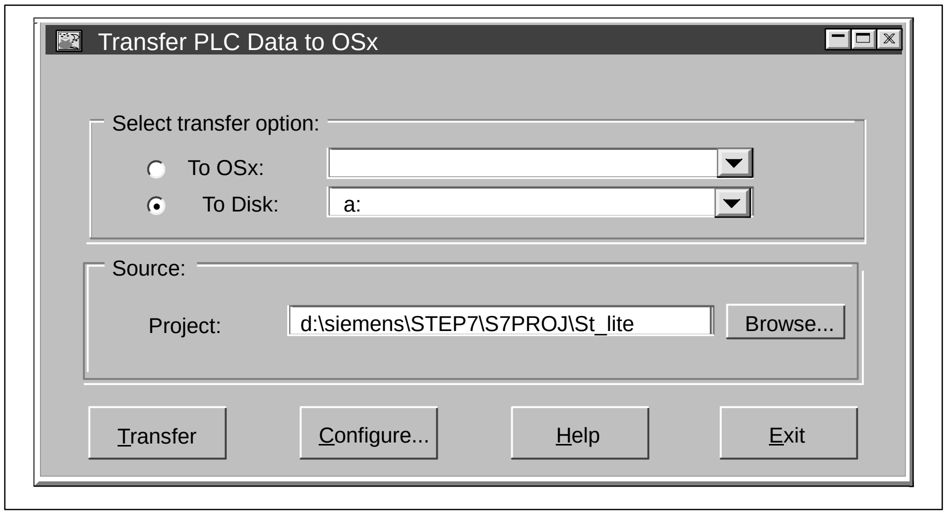
   To transfer to an OSx station, click in the circle to the left of **To OSx**.

6. Use the short-list display button to the right of the Destination field to set the destination.

   To transfer to disk, select the drive where your disk is located.

   To transfer to an OSx station, select the OSx station to which you are transferring the project. If you have not configured the OSx station in this utility, see Adding an OSx on page 2-44. The computer with the S7 Engineering Toolset installed must be connected to the same network as OSx. See page 1-19.

7. Click the **Transfer** button. The project displayed is transferred to the destination in a mapper file. If the project does not fit on a single disk, you will be prompted to insert another.



To OSx: [ ]
[✓ To Disk:] a:

Project: d:\siemens\STEP7\S7PROJ\St_lite

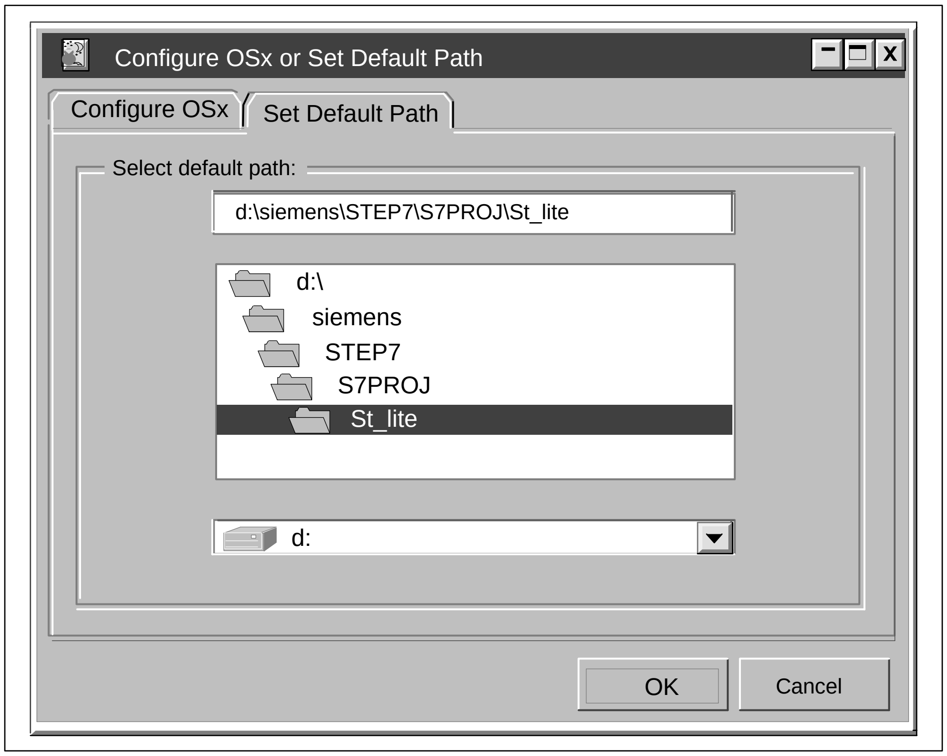
Figure 2-20 Transfer Dialog Box

Transferring the Mapper File to OSx (continued)

Setting a Default Path

To set a default path for a project mapper file, follow the steps below.

1. Click **Configure** on the Transfer dialog box. The Configure dialog box appears.
2. Select the **Set Default Path** tab. The Set Default Path dialog box appears.
3. In the field at the bottom of the dialog box, select the disk drive where the Engineering Toolset is installed.
4. Double-click on the appropriate directories to take you to your S7 project (**St_lite** in Figure 2-21).
5. Click **OK**. The path is now set to the default and will appear in the Path field of the Transfer dialog box.



d:\siemens\STEP7\S7PROJ\St_lite

Figure 2-21 Set Default Path Dialog Box

Adding an OSx Station

When you transfer the mapper file to an OSx station, it is recommended that you transfer to the primary, or to the sysadmin, if available. The primary will propagate the tags to all other OSx stations. The sysadmin will propagate the tags to the other OSx stations when it becomes the primary.

Before you can transfer the file to the OSx over the Industrial Ethernet, you must make sure that your network card on the engineering station is properly configured.

To add an OSx station to the destination list, follow the steps below.

1. Click **Configure** on the Transfer dialog box. The Configure dialog box appears (Figure 2-22).
2. Fill in the OSx Name and IP Address fields. (To find the exact name and address in OSx, select **Startup->Network Status** on the OSx main menu bar, highlight the OSx station, and click **View**.)
3. Click on the **Add** button, then click **OK**. The OSx station is added to the destination list.

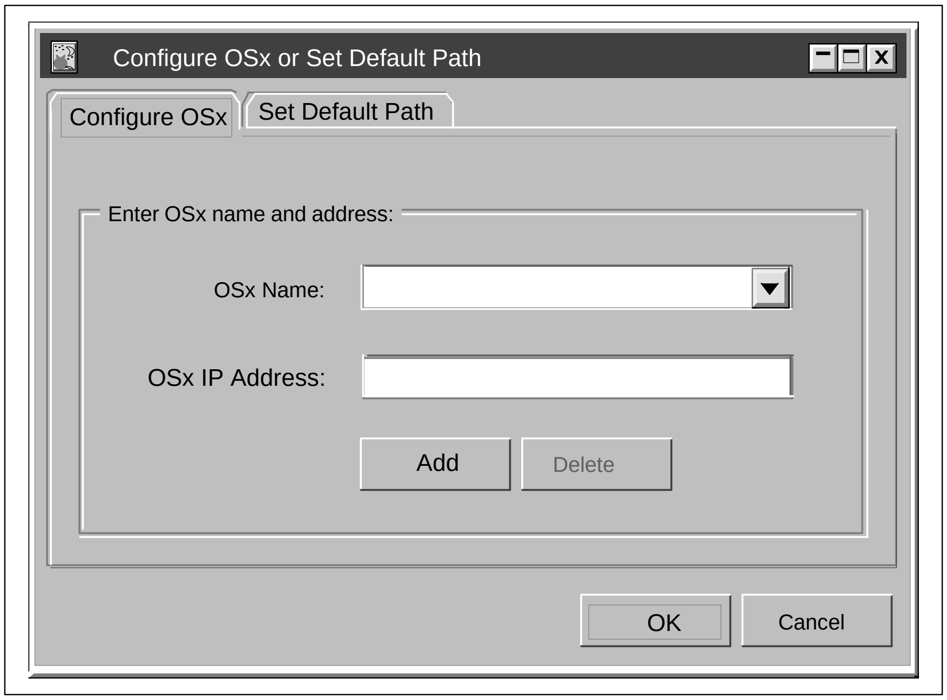
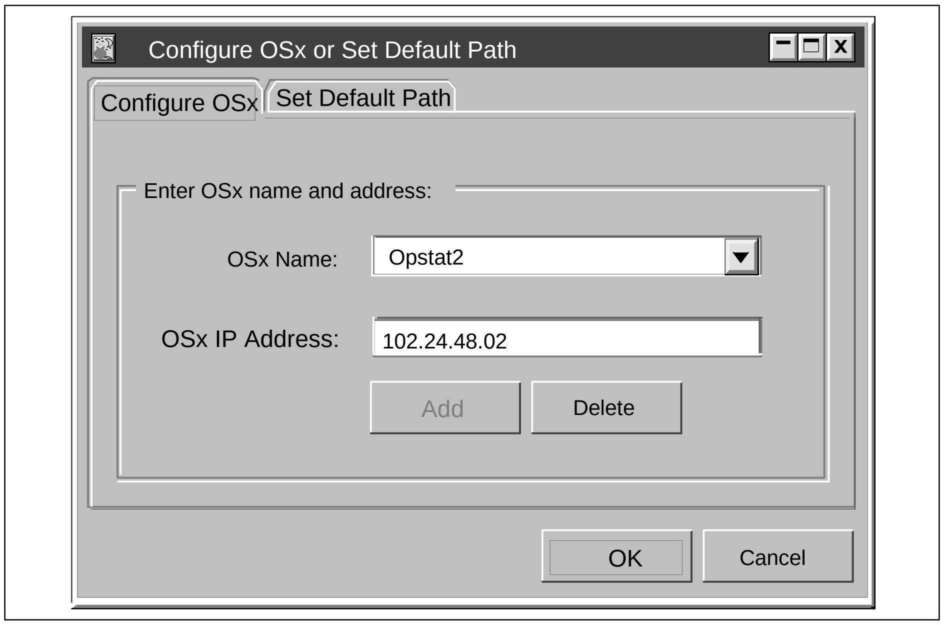


Figure 2-22 Configure OSx Dialog Box

Deleting an OSx Station

To delete an OSx station from the destination list, follow the steps below.

1. Click **Configure** on the Transfer dialog box. The Configure dialog box appears (Figure 2-22).
2. Select the name of the OSx station that you want to delete (Figure 2-23).
3. When the OSx name appears in the OSx Name field, click the **Delete** button, then click **OK**. The OSx station is deleted from the destination list.



Enter OSx name and address:

OSx Name: Opstat2

OSx IP Address: 102.24.48.02

Figure 2-23 Delete a Host

2.8 Installing S7 Tags in OSx

Installing Tags from the Mapper File in OSx

Once you have created the mapper file and transferred it to OSx or to a disk, you must install the S7 tags in OSx before another transfer occurs; otherwise the previous transfer is lost. Only the most recent transfer is installed.

If you want to check or modify the tags before you install them on OSx, you can convert the mapper file to one or more OSx tag data files and modify them directly (page 2-48).

When you install tags on the OSx station, it is strongly recommended that you do it in the Offline state. If you are installing the tags on the sysadmin, only the sysadmin needs to be in the Offline state.

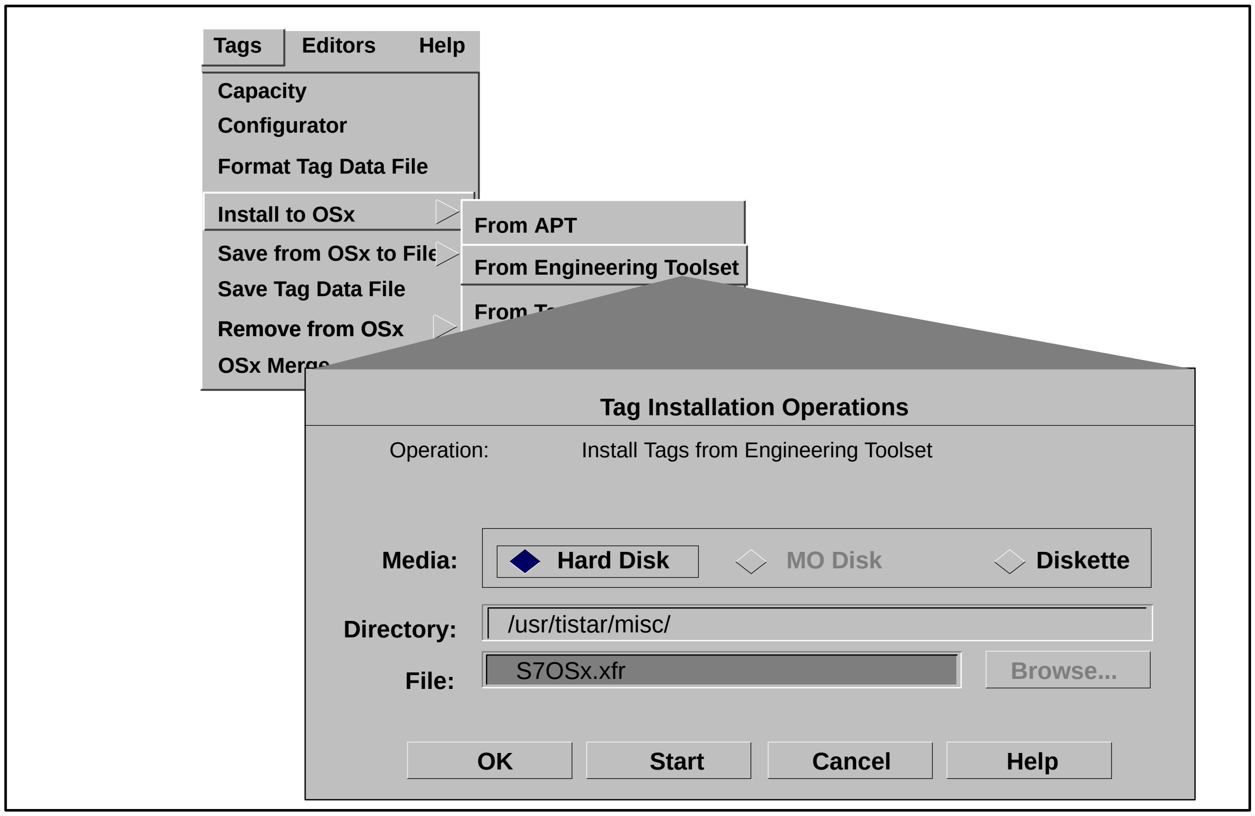
To install tags from a STEP 7 mapper file that is saved in the default mapper file directory (**/usr/tistar/misc**) and file (**S7OSx.xfr**), follow the steps below:

1. From the OSx main menu bar, select **Tags->Install to OSx->From Engineering Toolset**. The Tag Installation Operations dialog box appears (Figure 2-24).

2. If you are installing the STEP 7 mapper file from a diskette, click the button to the left of **Diskette**.

   If you are installing the STEP 7 mapper file from your hard drive, click the button to the left of **Hard Disk**.

3. Click the **Start** button to begin the tag installation. When installation is complete, the tag installation report is displayed. See Appendix A for an example of this report.



Operation: Install Tags from Engineering Toolset

Media: Hard Disk

Directory: /usr/tistar/misc/

File: S7OSx.xfr

Figure 2-24 Installing Tags from the Engineering Toolset

Converting the Mapper File to OSx Tag Data Files

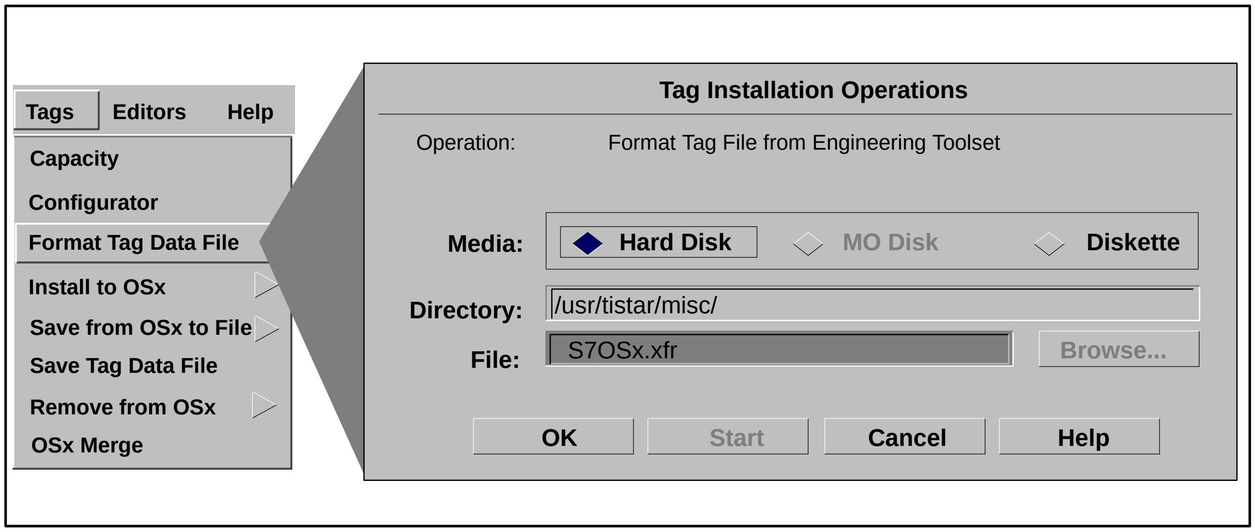
If you want to check the validity of the data transferred from the STEP 7 engineering station, you can convert a STEP 7 mapper file to an OSx tag data file before you install the tags in OSx. You can also make modifications in the tag data file, and note any new tag types that are represented in the mapper data.

Remember that any changes you make to the OSx tag data files will be overwritten the next time the mapper file is transferred from STEP 7. To convert the mapper file to OSx tag data files, follow these steps.

To convert a STEP 7 mapper file to an OSx tag data file, follow the steps below.

1. From the OSx main menu bar, select **Tags->Format Tag Data File**. The Tag Installation Operations dialog box appears (Figure 2-25).
2. If you want to read the STEP 7 mapper file from a diskette, click the button to the left of **Diskette**.

   If you want to read the STEP 7 mapper file from your hard drive, click the button to the left of **Hard Disk**.
3. Click **OK** to start the formatting process. The system generates a conversion log that notes new tag types and shows any errors or warnings. See Appendix D for explanations of conversion log error messages.



Operation: Format Tag File from Engineering Toolset

Media: [✓ Hard Disk] MO Disk Diskette

Directory: /usr/tistar/misc/

File: S7OSx.xfr

Figure 2-25 Format Tag File from Engineering Toolset

Installing Converted Files

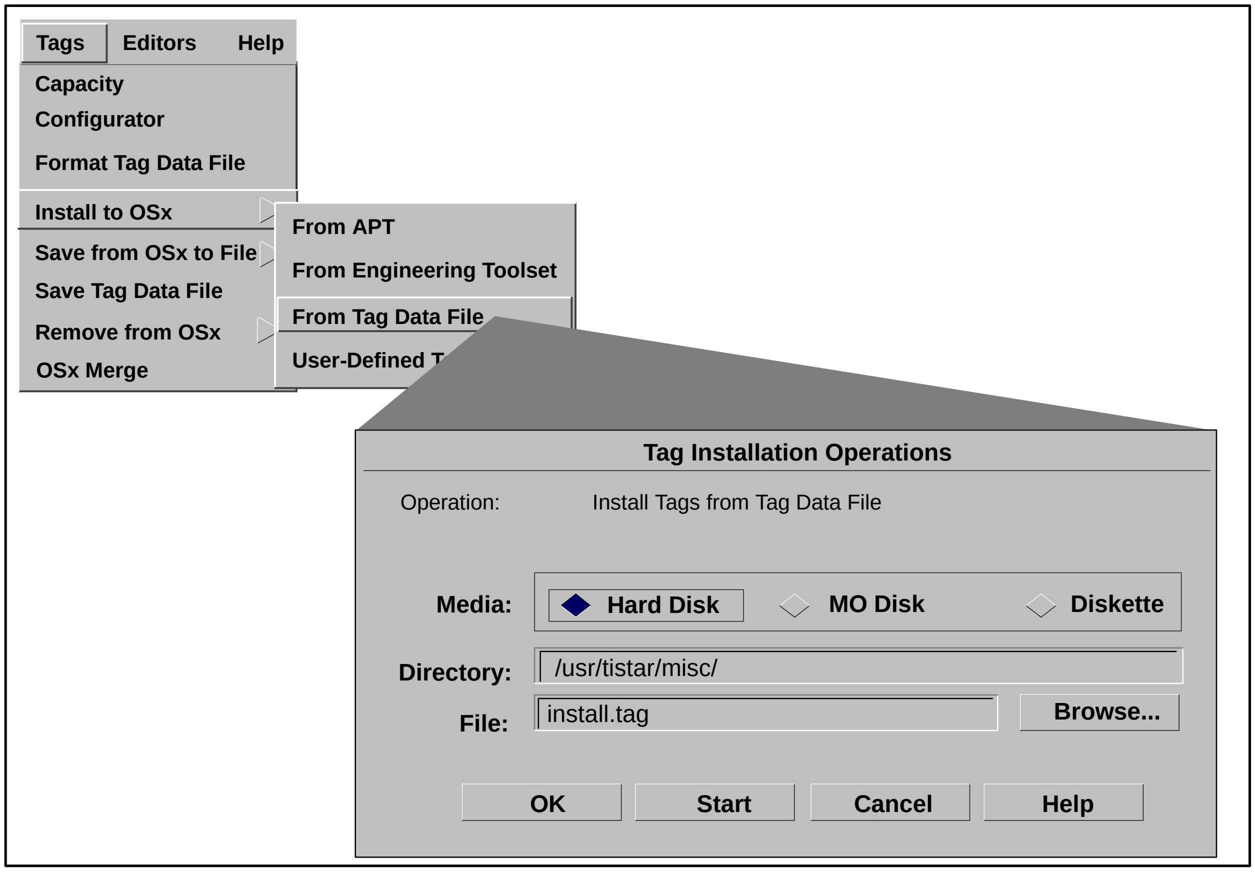
When you are ready to install the converted files, follow these steps:

1. From the OSx main menu bar, select **Tags->Install to OSx->From Tag Data File**. The Tag Installation Operations dialog box appears (Figure 2-26).

2. If you are installing the tag data file from your hard drive, click the button to the left of **Hard Disk**.

   If you are installing the tag data file from an MO disk, click the button to the left of **MO Disk**.

   If you are installing the tag data file from a diskette, click the button to the left of **Diskette**.



Operation: Install Tags from Tag Data File

Media: [✓ Hard Disk] MO Disk Diskette

Directory: /usr/tistar/misc/

File: install.tag

Figure 2-26 Installing Tags from Tag Data File

This procedure continues on the next page.

3. Click **Browse** to display the file selection list (Figure 2-27).

4. Use the file selection list to specify the directory path where your tag data file is located.

5. Click **OK** to confirm the directory path. The directory path and file name appear in the Tag Installation Operations dialog box.

6. In the Tag Installation Operations dialog box, click the **Start** button to begin the tag installation. When installation is complete, the tag installation report is displayed. See Appendix A for an example of this report.

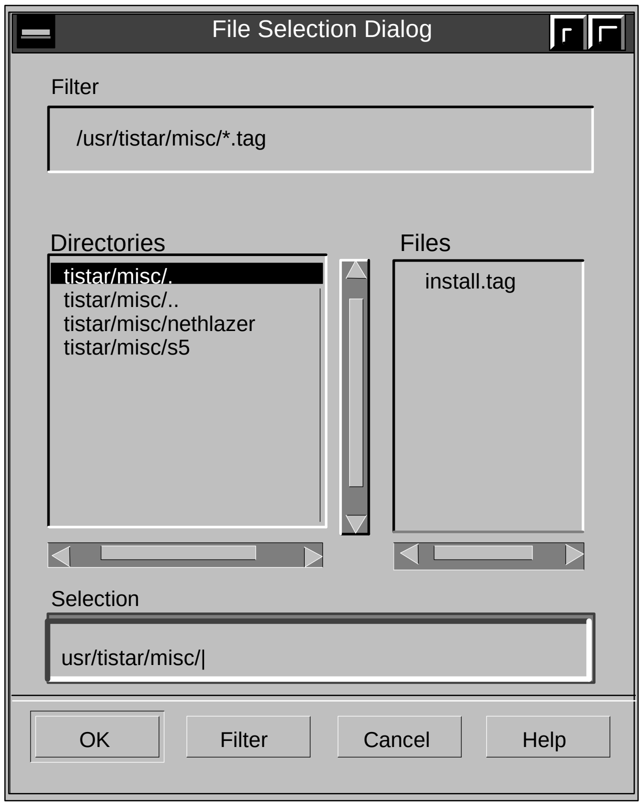


Figure 2-27 Selecting Directory Path for Tag Data File

Chapter 3
Creating a Sample Project

3.1 Preparing to Use the CFC

Overview

The OSx Library blocks are used in the CFC editor to control devices and relay information to the operator station. This chapter provides you with a sample program that uses the OSx Library blocks to control the filling, heating, and draining of a vessel.

Before you can begin the program example, you must complete the following tasks:

- Install the OSx Library programs in addition to the Engineering Toolset.
- Create your S7 project in the SIMATIC Manager.
- Configure the S7 CPU in the Hardware Configuration editor, as described in the STEP 7 documentation.

Adding a Charts Container

You need a Charts container in your S7 program to hold the CFCs and SFCs. To create a Charts container, follow these steps:

1. Expand the CPU folder by clicking on the + sign to the left of the icon. The S7 program folder appears.
2. Expand the S7 program folder. Two or three sub-folders appear. The Source Files folder and the Blocks folder appear. If the Charts folder does not appear, you must create it.
3. To create the Charts folder, highlight the S7 Program folder by clicking on it.
4. Click the right mouse button and select **Insert New Object->Chart container** from the menu. An icon named Charts appears under the S7 program folder. You will place your CFCs and SFCs in the Charts container.

Reserving Memory for OSx Functions

Function block numbers in the OSx library range from FB340 to FB407; functions range from FC900 to FC929. The CFC cannot use these ranges of numbers in compiling objects into new FBs and FCs; therefore, they must be disabled.

The default range for disabled functions in CFCs does not cover the OSx functions and function blocks. To ensure proper operation of the CFC with the OSx functions, you must reset the default for FC numbers. See the procedure on page 1-10 to reset this default value.

3.2 Process Requirements

Process Description

In the example process, a Fill Valve opens to allow liquid to enter the vessel. A temperature control valve is used to increase the temperature in the vessel to a specified setpoint. Once the setpoint temperature is reached, the temperature controller is set to manual mode and the temperature control valve closes. To drain the vessel, the Drain Valve opens and the Drain Pump starts. When the tank level reaches 5%, the Drain Pump stops and the Drain Valve closes.

Table 3-1 shows the initial conditions of the devices for the plant.

Table 3-1 Initial Conditions for the Process

Device Type	Device Name	Initial State
Drain Valve	V101	closed
Drain Pump	M101	stopped
Fill Valve	V102	closed
Agitator	M102	stopped
Temperature Controller	TIC101	manual, output = 0.0%
Note: All motors and pumps are single drive and single feedback. All valves are single drive and dual feedback.		

Modes of Operation

The process can operate in two modes:

- Manual: The operator has complete control.
- Automatic: The fill-heat-drain sequence repeats until manually halted.

Process Requirements (continued)

Sequential Events

Figure 3-1 shows the sequence of events for the example process. The detailed sequence is described below:

1. Open the Fill Valve.
2. Wait until the vessel level reaches the level setpoint, and then close the Fill Valve.
3. Place the Temperature Control loop in Automatic and set the temperature setpoint to the required value.
4. Once the required temperature is reached, put the Temperature Control loop in manual and set the output to 0.0%.
5. Open the Drain valve.
6. Once the Drain Valve is open, start the Drain Pump.
7. When level in the vessel is less than 5.0%, stop the Drain Pump and close the Drain Valve.

Process Interlocks

To ensure safe operation of the process, the design specification provides three interlocks:

- The Drain Pump operates only when the Drain Valve is open.
- The Agitator starts when the vessel level is greater than 20% and stops when the level is less than 20%.
- The Fill Valve closes when the vessel level is greater than 95%.

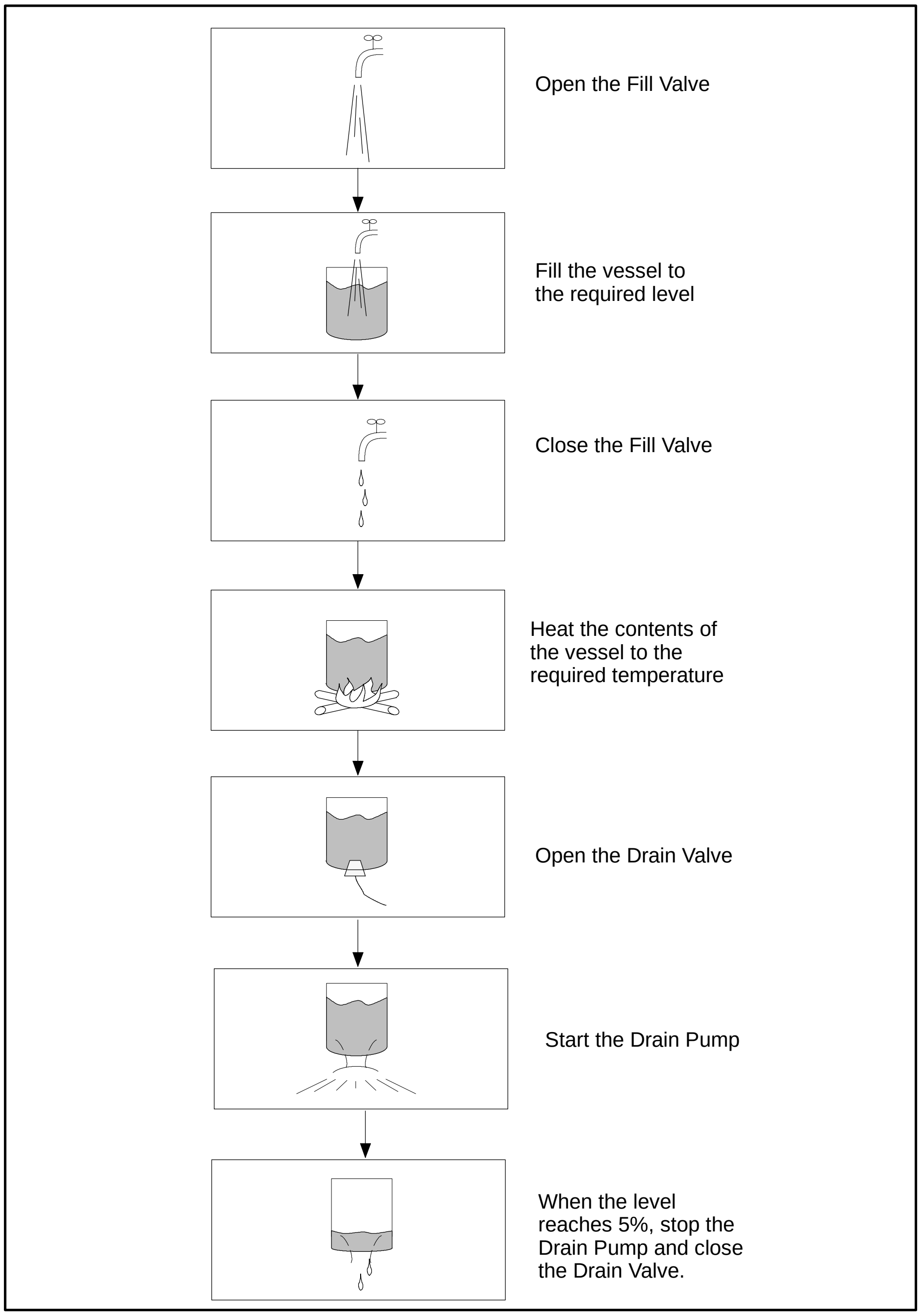

Open the Fill Valve
Fill the vessel to the required level
Close the Fill Valve
Heat the contents of the vessel to the required temperature
Open the Drain Valve
Start the Drain Pump
When the level reaches 5%, stop the Drain Pump and close the Drain Valve.

Figure 3-1 Sequence of Process Events

3.3 Implementing the Process

Using the process requirements described in Section 3.2, you can now implement the example process in an S7 controller.

Implementation uses the ”bottom-up” approach, beginning at the lowest point of automation. As each level of automation is complete the next level is added. Functions, objects, and automation are added gradually until the required process automation is reached.

Renaming the S7 Program

To begin the implementation, open the S7 project in the SIMATIC Manager. For tags to be installed in OSx, the name of the S7 program must be identical to the control node name that you configured in Network Setup on OSx. For example, if your control node name is CN4, follow these steps:

1. Expand the folders by clicking on the + icon until you see the S7 program under the CPU.
2. Click on **S7 Program(1)** to highlight it.
3. Click the right mouse button and select **Object Properties**.
4. In the Name field of the Properties – S7 Program dialog box, type `CN4`.
5. Select **OK**. The new **CN4** program name appears attached to the CPU.

Adding Required Blocks

In order for the OSx Library blocks to function properly, certain other function blocks must be present in the Blocks folder of your S7 program. It is recommended that you add these required blocks at the beginning of your implementation, although you can also add them later. To add required blocks, follow these steps:

1. Consult the *SIMATIC PCS 7 OSx Library Manual* for the device types that you plan to use in your process. The example program uses MSS and VSD. Find the Required Blocks heading under each device. The required blocks for both devices are listed below. These blocks need to be added to the Blocks folder only once.

   PACKSTAT (FC930)
   RBE_S (FB395)
   ALARM_S (SFC18)
   RD_SINFO (SFC6)

2. Under **CN4**, click on the **Blocks** folder to highlight it.
3. From the SIMATIC Manager menu bar, select **File->Open**. The Open dialog box appears.
4. In the Open dialog box, click in the circle beside **Libraries** to select it.
5. Double-click on **OSx Library** to select it. The OSx Library appears.
6. Expand the folders by clicking on the + icon until you see the **Blocks** folder. Click on **Blocks** to display the contents.
7. Arrange the windows on your screen so that both the OSx Library and the CN4 Program Blocks folders are visible.
8. In the OSx Library Blocks window, highlight the four required blocks by simultaneously holding down the **Ctrl** key and clicking the left mouse button.
9. After all the blocks are highlighted, click the left mouse button over one of the selected blocks, and hold the mouse button down to drag and drop all four blocks into the Blocks folder of the CN4 Program. When the blocks are placed in the Blocks folder, their names and function block numbers automatically appear in the Symbol Table.

Implementing the Process (continued)

Inserting the CFC

On page 3-2, you created a Charts container for the example S7 program. To add a CFC for the OSx Library function blocks, follow the steps below.

1. Click on the **Charts** container. At this point, it is empty.
2. To add a CFC, click the right mouse button on **Charts** and select **Insert New Object–>CFC** from the menu. This creates an empty CFC chart.
3. Rename the new CFC by typing `Vessel` over the **CFC1** name that appears highlighted in a box.

Placing Function Blocks in the CFC

To control the field devices, you must place function blocks in the CFC. The example starts with the Fill Valve and the Agitator Motor. From the design criteria you know that the valve is a VSD type and the motor is an MSS type. Follow the steps below to add these blocks to the CFC editor:

1. Double-click on the CFC **Vessel** to open it in the CFC editor. There are six sheets available for editing.
2. Double-click in the upper left sheet to zoom in on it. (You can double-click in any white area of the CFC to zoom out again.)
3. Select **View–>Catalog** from the CFC menu bar. A catalog of function blocks appears to the right of the CFC chart.
4. Expand **Libraries–>OSx Library–>S7 Program(1)** in the catalog to display the OSx Library function blocks.
5. Scroll down to locate the VSD valve (FB345), click, and drag it into the CFC chart. If the required blocks for this device are not present, the function block will not be placed in the CFC. You must place these required blocks in the Blocks folder first (page 3-7).
6. Scroll down to locate the MSS valve (FB363), click, and drag it into the CFC chart. The required blocks are already present in the Blocks folder, since they are the same as for the VSD valve.

Naming the Function Blocks in the CFC

Once the function blocks are placed in the CFC, you can configure their tag names and descriptions. Follow these steps:

1. Click the right mouse button on the title bar of the VSD function block.
2. Select **Object Properties** from the menu.
3. In the General folder of the Block Object Properties dialog box, rename the block and give it a meaningful description. Type `V102` in the Name field, and change the description in the Comment field to `Fill Valve`.
4. Click **OK**. Now look at the block. Note that the name in the upper left corner is V102 and the description is Fill Valve. This description is for the CFC editor only.
5. To give the block a description that will translate to OSx, open the Object Properties dialog box again, and click on the **Operator Control & Monitoring** button.
6. In the Comment field of the Operator Control & Monitoring dialog box, enter the description `Fill Valve V102`. For a complete description of the parameters that you can configure in this field, see page 2-11.
7. Click **OK** to save this change, and click **OK** again to exit the Object Properties dialog box.
8. Repeat steps `1-7` for the MSS function block. Enter the name `M102` in the Name field of the General folder of the Block Object Properties dialog box, and `Agitator Motor` as the description in the Comment field. Then in the Comment field of the Operator Control & Monitoring dialog box, enter `Agitator Motor M102` as the description.

Implementing the Process (continued)

Changing Block I/O Properties

Now that you have placed the blocks in the CFC editor, the alarm times for the devices need to be modified. If you have only one or two I/O changes to make, you can choose the following procedure. Follow the steps below to change the timers for the V102 device.

1. Double-click on the **O_ALRM_T** input on the V102 function block. The Properties – I/O dialog box appears.
2. In the Value field, enter `5.0`, and click **OK**. This changes the open alarm timer for the device.
3. Repeat steps 1 and 2 for the **C_ALRM_T** input to change the close alarm timer to 5 seconds.

If you have several I/O changes to make, you can make them all at once. Follow the steps below to change the timers for the M102 device.

1. Double-click on the title bar on the M102 function block. The Block Object Properties dialog box appears.
2. Select the **I/Os** tab, and scroll down to locate the **R_ALRM_T** and **S_ALRM_T** inputs.
3. In the Value column, enter `5.0` for both inputs, and click **OK**. This changes the run and stop alarm timers for the device to 5 seconds.

SAMPLE_T

The OSx Library function blocks are placed by default in OB35, which executes once every 100ms. The default for **SAMPLE_T** is 0.1 seconds. This means that every time the block executes, the alarm timers decrease by 0.1 seconds. Since the time base is in seconds, you do not need to change **SAMPLE_T** unless you place the function block in another OB with a different cycle time. See the *SIMATIC PCS 7 OSx Library Manual* for a detailed discussion on using **SAMPLE_T**.

Copying the Function Blocks in the CFC

The CFC now contains a VSD device Fill Valve and an MSS device Agitator Motor. The next task is to create the Drain Valve and the Drain Pump. Although you can drag and drop more of the same devices into the CFC, you can also copy the devices you have already configured. Follow the steps below:

1. Right click on the title bar of the V102 function block.
2. Select **Copy** from the menu.
3. Move the cursor into a white space on the CFC, right click again, and select **Paste**.
4. When the hand cursor appears, place it over the function block with the dotted outline, click, and drag the copy to a clear space in the CFC. A new VSD function block appears with the same I/O properties as V102.
5. Follow steps 1–7 of the procedure on page 3-9 for the copied VSD function block. Enter the name `V101` in the Name field of the General folder of the Block Object Properties dialog box, and `Drain Valve` as the description in the Comment field. Then in the Comment field of the Operator Control & Monitoring dialog box, enter `Drain Valve V101` as the description.

For the Drain Pump, copy the M102 function block following the steps above. Enter the name `M101` in the Name field of the General folder of the Block Object Properties dialog box, and `Drain Pump` as the description in the Comment field. Then in the Comment field of the Operator Control & Monitoring dialog box, enter `Drain Pump M101` as the description.

Inserting an Alarm Block

The example process uses an ALRM function block (FB384) to monitor the level of fluid in the vessel. Follow the steps below.

1. When you check the *SIMATIC PCS 7 OSx Library Manual* for the required blocks for this function block, you see that five additional blocks are required. Note that two of the blocks (FC930 and SFC6) were used by the devices. You do not need to load these two blocks again. Load the remaining three blocks listed below, following the procedure on page 3-7.

   ADAPTER (FB393)
   RBE_P (FB394)
   ASC_ALRM (FB396)

2. In the CFC editor, scroll down in the OSx Library catalog to locate the ALRM function block.

3. Drag and drop the ALRM block into the CFC sheet.

4. Edit the Block Object Properties for the ALRM function block following the procedure on page 3-9. Enter the name `LI101` in the Name field of the General folder of the Block Object Properties dialog box, and `Vessel Level` as the description in the Comment field. Then in the Comment field of the Operator Control & Monitoring dialog box, enter `Vessel Level LI101` as the description.

5. To set initial values for the ALRM block I/O, double-click on the title bar on the LI101 function block. The Block Object Properties dialog box appears.

6. Select the **I/Os** tab, and scroll down to locate the following inputs, and enter the initial values given below in the Value column.

ENABL	=	1
PVH	=	100.0
HHA	=	94.0
HA	=	90.0
MDEV_YO	=	0

   Be sure to specify all alarm limits, since the default settings may not apply to your application. By doing this, you also eliminate nuisance alarms from the system.

7. Click **OK**. This changes the LI101 inputs to the new initial values.

Associated Math

Even though there is no associated math for the LI101 ALRM block, you must still have a connection for the **ASO_FB** input. See the discussion on associated math under the ALRM function block in the *SIMATIC PCS 7 OSx Library Manual*.

1. Right mouse click on the **ASO_FB** input.
2. Select **Insert connection to operand**.
3. Type in `ASC_ALRM` in the Symbol/Operand field and click **OK**.

   The ASC_ALRM function block exists in the Symbol Table because you placed the required function blocks in the Blocks folder of the program.

Adding the Temperature Controller

The example process uses a PID function block (FB382) to control the temperature of the vessel. Follow the steps below.

1. When you check the *SIMATIC PCS 7 OSx Library Manual* for the required blocks for this function block, you see that five additional blocks are required. Note that three of the blocks are already in the Blocks folder. You do not need to load these blocks again. Load the remaining two blocks (SFB35 and FB397) following the procedure on page 3-7.

   ASC_PID (FB397)
   ALARM_8P (SFB35)

2. In the CFC editor, scroll down in the OSx Library catalog to locate the PID function block.

3. Drag and drop the PID block into the CFC sheet.

4. Edit the Block Object Properties for the PID function block following the procedure on page 3-9. Enter the name `TIC101` in the Name field of the General folder of the Block Object Properties dialog box, and `Temperature Control Loop` as the description in the Comment field. Then in the Comment field of the Operator Control & Monitoring dialog box, enter `Temperature Control Loop TIC101` as the description.

5. To set initial values for the PID block I/O, double-click on the title bar on the TIC101 function block. The Block Object Properties dialog box appears.

6. Select the **I/Os** tab, and scroll down to locate the following inputs, and enter the initial values given below in the Value column.

SPH	=	100.0
SPL	=	0.0
PVH	=	300.0
PVL	=	0.0
HHA	=	94.0
HA	=	90.0
KC	=	0.1
TI	=	10.0
MDEV_YO	=	0

   Be sure to specify all alarm limits, since the default settings may not apply to your application. By doing this, you also eliminate nuisance alarms from the system.

7. Click **OK**. This changes the TIC101 inputs to the new initial values.

8. Even though there is no associated math for the TIC101 PID block, you must still have a connection for the **ASO_FB** input. Follow the procedure on page 3-13, and type `ASC_PID` in the Symbol/Operand field.

   The ASC_PID function block exists in the Symbol Table because you placed the required function blocks in the Blocks folder of the program.

3.4 Adding Simulation Code to the Process

Creating a Simulation Code Function Block

All the devices for the example process have been created. The next step is to add simulation to each of the devices to reflect the action of the process in the field. Using the simulation, you can verify that the controller program operates as required.

You write simulation code to perform the functions needed to create the process. In this example, a custom block performs the simulation. The code can be modified to fit various applications. A function block (FB200 in this example) containing the simulation code is created in SCL and shown in Appendix B.

To create the simulation code for this process, follow these steps:

1. In the SIMATIC Manager, select **Source Files** under the CN4 program folder.
2. Right mouse click and select **Insert New Object->SCL Source File**. A new icon labeled **SCL Source File(1)** appears in the Source Files folder.
3. Rename the new source file by typing `Vessel Sim` over the **SCL Source File(1)** name that appears highlighted in a box.
4. Double-click on the **Vessel Sim** source file to open it.
5. Select **Insert->Block Template->FB** from the menu bar. A block template for the function block appears in the source file. Fill in the blanks, using the SCL simulation code from Appendix B.
6. Select **File->Save** from the menu bar to save the SCL simulation code.
7. Select **File->Compile** from the menu bar to compile the simulation code and create the function block (in this case, FB200) in the Blocks folder.
8. Select **PLC->Download** from the menu bar to load the function block into the controller. Perform testing to verify that your simulation code functions as intended.

Importing an External Source File

If you have already written a simulation block that you want to use here, or if you have created the source file in another text editor, you can import the source file from disk or another directory. Follow these steps:

1. In the SIMATIC Manager, select **Source Files** under the CN4 program folder.
2. Right mouse click and select **Insert New Object–>External Source File**. The Insert External Source File dialog box appears.
3. In the **Look in** field, use the down-pointing arrow to select the drive where your source file is located.
4. Double-click on the folder where your source file is located to open it.
5. Double-click on the source file to place it in the Source Files folder.
6. Double-click on the source file to open it.
7. Select **File–>Compile** from the menu bar to compile the simulation code and create the function block in the Blocks folder.
8. Select **PLC–>Download** from the menu bar to load the function block into the controller. Perform testing to verify that your simulation code functions as intended.

Adding Simulation Code to the Process (continued)

Placing the Simulation Block in the CFC

Once the simulation block has been compiled and tested, it needs to be connected to the devices. To connect the device function blocks to the simulation, the simulation function block must be placed in the CFC. Follow the steps below.

1. Open the Vessel CFC.
2. Expand the CN4 catalog in the right-hand window. FB200 now appears in the list of blocks for the CN4 program.

   If FB200 does not appear in the list, select **View–>Update** from the menu bar to display it.
3. Click on **FB200** and drag and drop it from the Block Catalog to the CFC sheet.

Displaying Multiple Windows of the Same CFC

Since the simulation block is in a different sheet of the CFC from the devices, connections must be made across sheets. To display two sheets of the CFC at the same time, follow the steps below.

1. Select **Window–>New Window** from the menu bar. This opens a second window into the same CFC.
2. Select **Window–>Arrange–>Vertically** from the menu bar. This tiles the windows side by side.
3. In the left-hand window, double-click in a white area of the sheet where the devices are located.
4. In the right-hand window, double-click in a white area of the sheet where the simulation block is located. Both windows should now be in single-sheet view. (If not, double-click again.)

Connecting Blocks in Different Sheets of the CFC

Now that you have two windows in single-sheet view, you need to connect the devices to the simulation block. First, connect the **CMMD** output for V101 Drain Valve to the **V1_OP_CL** input of the FB200 simulation block. Follow the steps below.

1. Using the scroll bars, center the left-hand sheet on V101 and the right-hand sheet on FB200.

2. Click on the **CMMD** output on the V101 function block. The output highlights in dark blue.

3. Click on **V1_OP_CL** input of the FB200 simulation block. A line appears connecting the V101 block to the right-hand margin. Another line appears connecting the FB200 block to the left-hand margin. Text appears in the margin identifying the connection to the other block.

   If the line does not connect, double-click on the output. In the Properties – I/O dialog box, click **Signal** in the Import-Export Assistant field. Click **OK** and retry steps 2 and 3.

4. Repeat steps 2 and 3 for each block connection listed in Table 3-2. Center the left-hand sheet on V102, M101, or M102 as necessary (Figure 3-2 and Figure 3-3).

Table 3-2 Block Interconnections

Output Block	Output	Input Block	Input
V101	CMMD	FB200	V1_OP_CL
V102	CMMD	FB200	V2_OP_CL
FB200	FB_V1_OP	V101	OLS
FB200	FB_V1_CL	V101	CLS
FB200	FB_V2_OP	V102	OLS
FB200	FB_V2_CL	V102	CLS
M101	CMMD	FB200	M1_ON_OF
M102	CMMD	FB200	M2_ON_OF
FB200	FB_M1_ON	M101	RUNIO
FB200	FB_M2_ON	M102	RUNIO

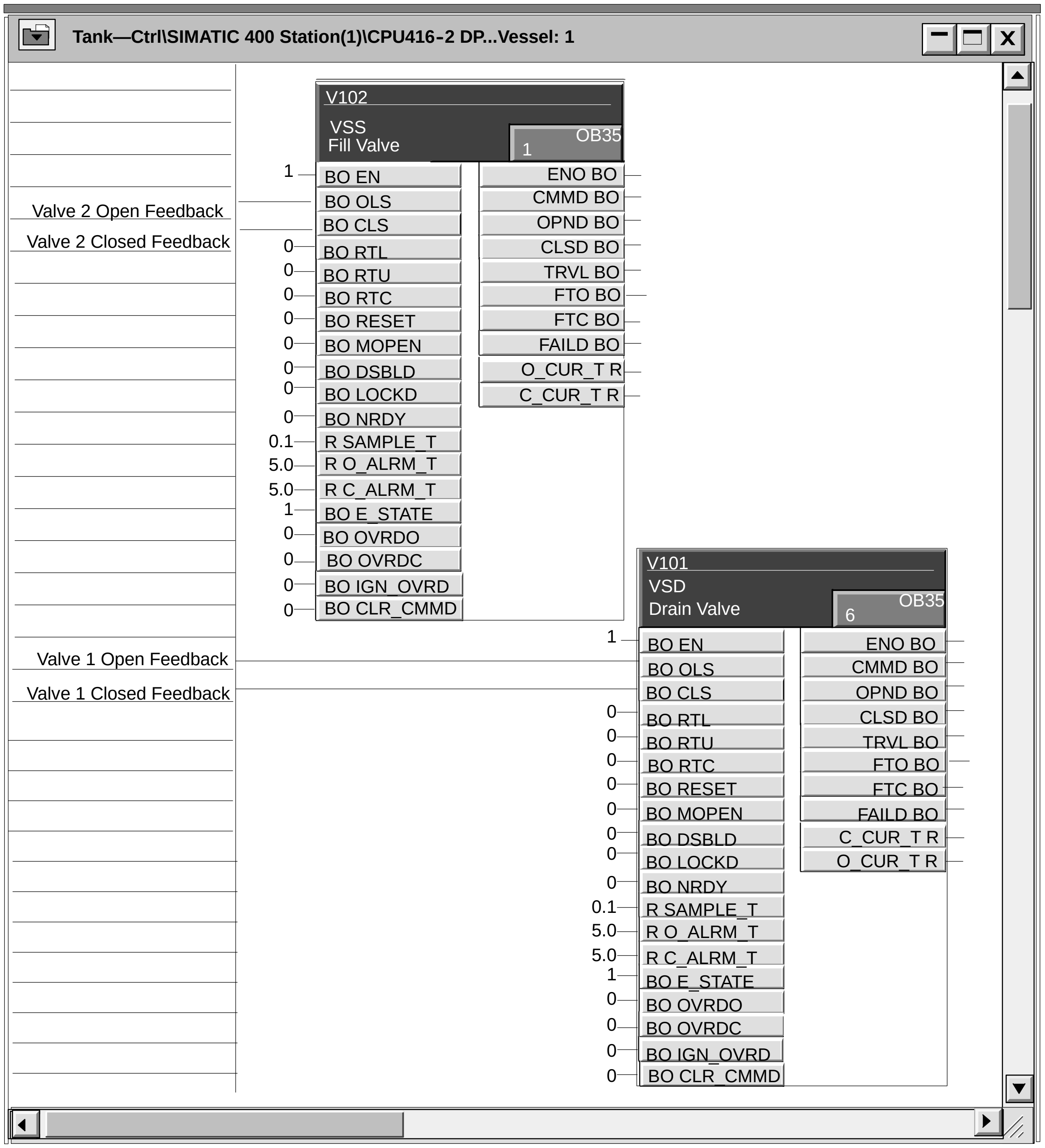


Figure 3-2 Left-Hand Window: Valve Function Block Connections

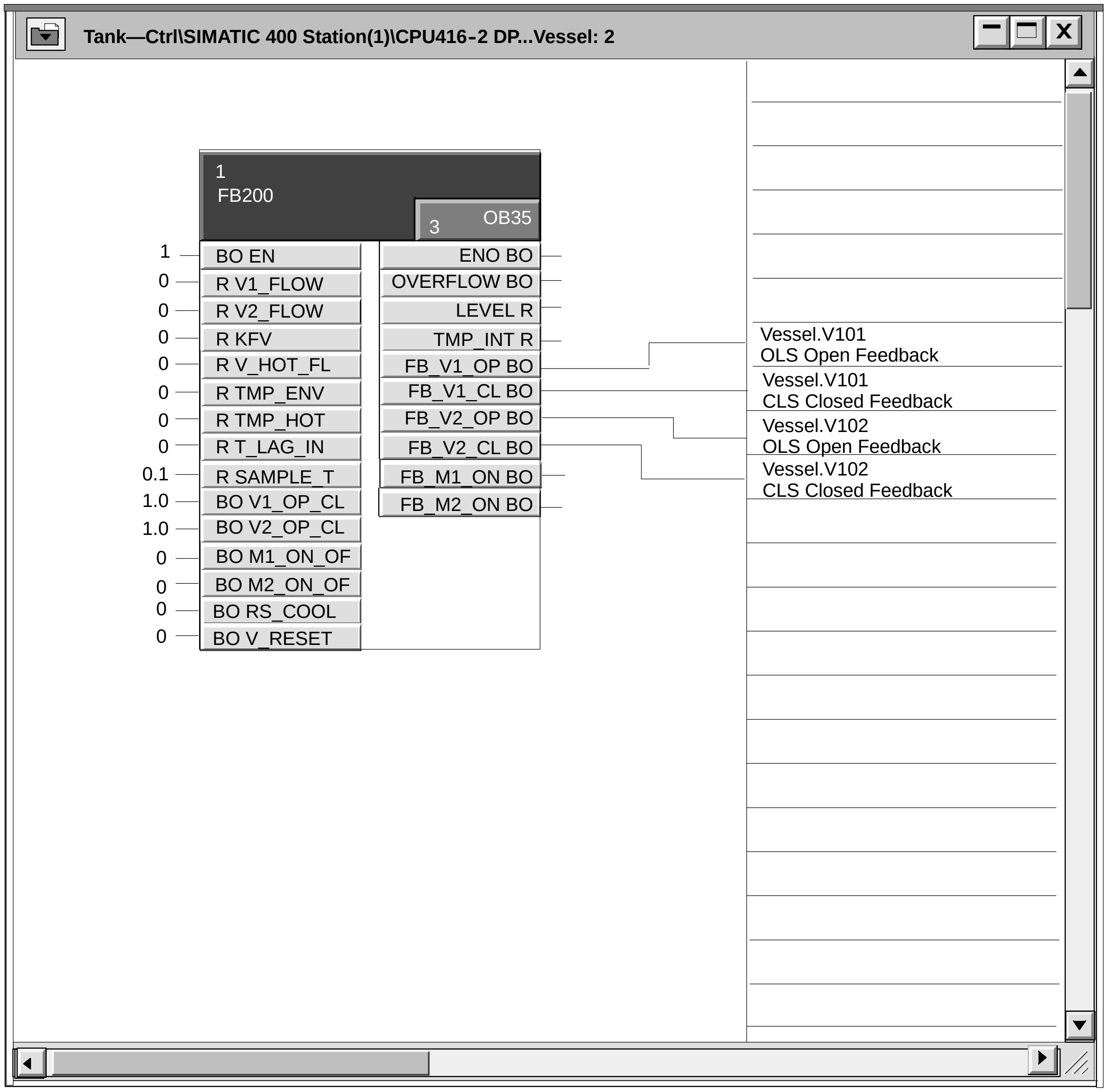


Figure 3-3 Right-Hand Window: Simulation Block Connections

Adding Simulation Code to the Process (continued)

Connecting Blocks in the Same Sheet of the CFC

The simulation function block FB200 increases the level of the vessel if the Fill Valve is open, and decreases the level if the Drain Valve is open. The **LMN** output of the PID loop is sent through a lag function to simulate gradual heating. This temperature is then returned to the PID loop as the process variable **PV**.

To model the change of level in the vessel, you must connect the **LEVEL** output of FB200 to the **PV** input of the LI101 function block. To simulate the temperature change in the vessel, you must make two connections from FB200 to the TIC101 function block. Follow the steps below:

1. In the CFC editor, go to the single-sheet view of the sheet with the simulation function block, the ALRM block, and the PID block.
2. Click on the **LEVEL** output of FB200 and hold the left mouse button down. The output highlights in dark blue.
3. Drag the cursor to the **PV** input of the LI101 function block and release the mouse button. A line appears connecting the output of one block to the input of the other.
4. Repeat steps 2 and 3 to connect the **LMN** output of the TIC101 block to the **V_HOT_FL** input of FB200, and to connect the **TMP_INT** output of FB200 to the **PV** input of the TIC101 block. Lines appear connecting the outputs to the inputs.

NOTE: As you add code to your project, you should compile, download and test frequently to insure the proper operation of your code. If you write too much code before testing, it becomes harder to debug. See page 3-24 for compiling and downloading your CFC.

Using a Standard S7 Block

To ensure safe operation of the process, the design specification provides three interlocks:

- The Agitator operates only when the vessel level is greater than 20%.
- The Fill Valve closes when the vessel level is greater than 95%.
- The Drain Pump operates only when the Drain Valve is open.

To control the Agitator, you use the standard S7 block CMP_R to compare the vessel level to a preset value, in this example, 20.0. Follow these steps.

1. In the CFC editor, click to expand the **ELEMENTA** folder which is located under **Libraries->CFCLIBS**.
2. Click on the CMP_R (compare real) function block, drag, and drop in an empty sheet of the CFC.
3. Edit Block Object Properties following the procedure on page 3-9 to name the block `Level Compare` and give it the following description in the Comment field: `Level Compare for Agitator`. Because this block is not used as a tag in OSx, you do not need to enter a description in the Operator Control and Monitoring dialog box.
4. Connect the **LEVEL** output of FB200 to the **IN1** input of the Level Compare block following the procedure for displaying multiple windows on page 3-18 and for connecting blocks in different sheets on page 3-19.
5. Double-click on the **IN2** input of the Level Compare block to open the Properties – I/O dialog box.
6. Enter `20.0` in the Value field and click **OK**.
7. Connect the **GT** output of the Level Compare block to the **RTR** input of the M102 Agitator block.
8. Connect the **LT** output of the Level Compare block to the **RTS** input of the M102 Agitator block.

When the level in the vessel is greater than 20%, the Agitator is requested to start. When the level in the vessel falls below 20%, the Agitator is requested to stop.

Adding Simulation Code to the Process (continued)

To implement the interlock for the Fill Valve, you use another CMP_R block to compare the vessel level to a preset value, in this example, 95.0. Follow these steps.

1. Copy the Level Compare (CMP_R) block following the procedure on page 3-11 for copying function blocks. Enter the name `Fill Compare` and the description `Level Compare for Fill Valve`.
2. Connect the **GT** output of the Fill Compare block to the **NRDY** input of the V102 Fill Valve block.
3. Connect the **LEVEL** output of FB200 to the **IN1** input of the Fill Compare block.
4. Double-click on the **IN2** input of the Fill Compare block to open the Properties – I/O dialog box.
5. Enter `95.0` in the Value field and click **OK**.

When the level in the vessel is greater than 95% the NRDY input of the Fill Valve will be true and the valve will close.

To control the Drain Pump, connect the **CLSD** output of V101 to the **NRDY** input of the M101 block. When the Drain Valve is closed the **NRDY** input of the Drain Pump is true, preventing the operation of the Drain Pump.

Compiling and Downloading the CFC

To compile and download the CFC, follow the steps below.

1. Select **Chart->Compile** from the menu bar of the CFC editor. The Compiling dialog box appears.
2. Select **Complete Program** and click **OK**. The program compiles. This process may take several seconds.
3. If errors occur, fix them and recompile the program. Otherwise, click **OK**.
4. Select **PLC->Download** from the menu bar of the CFC editor. The program downloads to the controller.
5. Debug the program until your process requirements are met.

3.5 Using a Symbol in the Control Process

Creating a Symbol

To start the sequential part of the process automatically, create a symbol to be used in the SFC.

1. In the standard hierarchy of the SIMATIC Manager, select the **CN4** program to highlight it.
2. In the right-hand side of the window, double-click on the **Symbols** icon to open the Symbol Editor.
3. In an empty row of the Symbol Table, enter `auto_cycleYC`. The `YC` suffix indicates how the symbol is saved in the OSx database. See Table 2-5 on page 2-24.
4. Enter `M0.1` in the Address column. The data type defaults as boolean.
5. In the Comment column, enter `Continue Sequence`.
6. Click on the symbol name to highlight it.
7. Click the right mouse button and select **Special Object Properties-> Operator Control and Monitoring**.
8. Click in the box to the left of Operator Control and Monitoring at the top of the dialog box. A check appears in the box, marking the symbol for translation to OSx.
9. In the Comment field, enter `Auto Cycle` for the description to be entered in the OSx database, and click **OK**.
10. Select **Symbol Table->Save** from the Symbol Editor menu bar to save your symbol.
11. Select **Symbol Table->Exit** from the Symbol Editor menu bar to exit.

3.6 Using an SFC for Sequential Control

Overview of the SFC

You have now completed all automation for the continuous functions of the process. You must create an SFC to perform the following sequential control steps: FILL, HEAT, and DRAIN. Figure 3-4 shows the completed SFC Chart for Ves_Ctrl. Follow the procedures in this section to create the SFC.

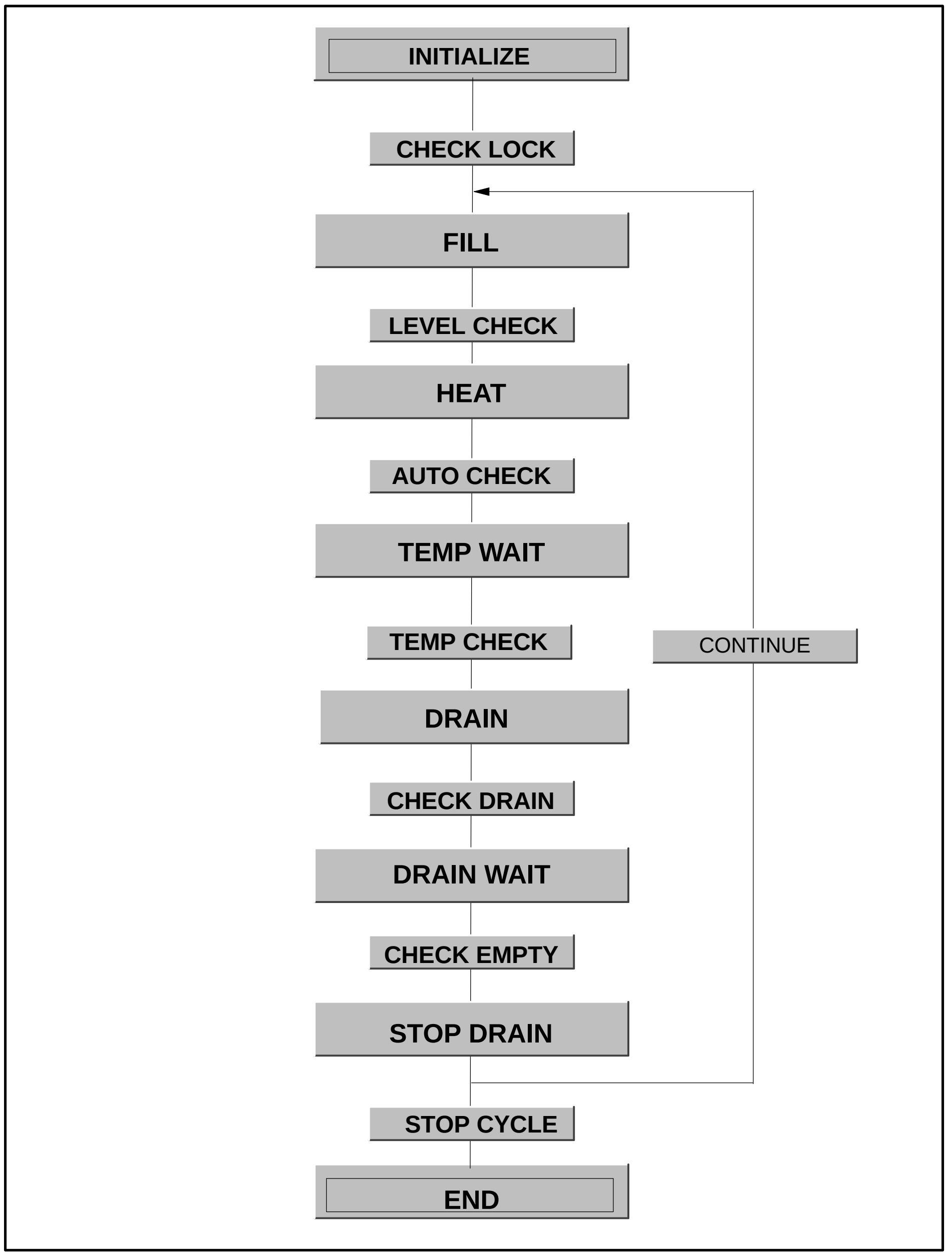


Figure 3-4 Ves_Ctrl SFC

Inserting the SFC

On page 3-2, you created a Charts container for the example S7 program. To add an SFC for the sequential control of the process, follow the steps below.

1. Click on the **Charts** container.
2. To insert an SFC, click the right mouse button on **Charts** and select **Insert New Object->SFC** from the menu. This creates an empty SFC chart.
3. Rename the new SFC by typing `Ves_Ctrl` over the **SFC1** name that appears highlighted in a box.

Inserting Steps and Transitions in SFC

To insert steps and transitions for the Ves_Ctrl SFC, follow the steps below.

1. Double-click on the **Ves_Ctrl** icon to open it in the SFC editor. A START step and an END step appear with a transition between them.
2. Click on the **Insert step + transition** icon in the vertical toolbar on the left side of the screen. You can now insert boxes for steps and transitions.
3. Place the cursor on the sheet until a horizontal line appears indicating that a step and transition can be inserted at this point.
4. Click to insert an empty step and transition.
5. Continue inserting steps and transitions until you have six steps between START and END.

Naming Steps in SFC

To name the first step for the Ves_Ctrl SFC, follow the steps below:

1. Click on the **Select** icon on the SFC vertical toolbar.
2. Double-click on the START step. The Object Properties dialog box appears.
3. In the Name field of the General folder, enter `INITIALIZE` to rename the step.
4. In the Chart Comment field, enter `Initialize Devices to Auto Mode`.

Setting I/O Values for Steps in SFC

A small box above an input or output of the function block in CFC indicates that an SFC transition has read access to it; a small box below an input or output of the function block indicates that an SFC step has write access to it. The input or output is accessed by means of the *CFC_name.FB_name.ext*.

To set I/O values for the first step in the Ves_Ctrl SFC, follow the steps below.

1. Select the **Initialization** tab. The code that you enter here executes only once when the INITIALIZE step executes.
2. Click the **Browse** button at the bottom of the dialog box. The Browse dialog box appears.
3. Double-click on **Vessel** in the Charts column of the CFC Charts folder. A list of the blocks in the CFC appears in the Blocks column.
4. Double-click on **M101** block. A list of I/O points for the M101 function block appears in the I/Os column.
5. Double-click on the **RTL** input. The following text appears in the Initialization folder of the Object Properties dialog box.

   ```
   Vessel.M101.RTL
   ```

6. In the column to the right of the = sign, enter `1`, and click the cursor in the first column of the next row down. The word TRUE appears in the assignment field. When the INITIALIZE step executes, the RTL input is set to true.
7. Repeat steps 2–6 for the **RTL** inputs of V101, V102, and M102.
8. Click the **Apply** button to save the information.

Navigating between Steps in SFC

To configure the other steps in the Ves_Ctrl SFC, follow the steps below.

1. Use the down-arrow button at the bottom of the Object Properties dialog box to move to the next SFC step.
2. Repeat the procedures for naming steps and setting I/O values described above with the information in Table 3-3.

Table 3-3 I/O Values in Ves_Ctrl Steps

Step Name	Comment	Initialization Code

INITIALIZE	Initialize Devices to Auto Mode	Vessel.M101.RTL := TRUE
		Vessel.M102.RTL := TRUE
		Vessel.V101.RTL := TRUE
		Vessel.V102.RTL := TRUE

FILL	Open the Fill Valve	Vessel.V102.RTO := TRUE

HEAT	Put Temp Controller in Auto
	Close Fill Valve
		Vessel.TIC101.RATO := TRUE
		Vessel.V102.RTC := TRUE

TEMP WAIT	Set the setpoint of the loop	Vessel.TIC101.SP := 100.0

DRAIN	Open the Drain Valve
	Turn on Drain Pump
	Put TIC in Manual & 0.0 out
		Vessel.V101.RTO := TRUE
		Vessel.M102.RTR := TRUE
		Vessel.TIC101.RMAN := TRUE
		Vessel.TIC101.OUT := 0.0

DRAIN WAIT	Waiting for the level to reach specified level.	

STOP DRAIN	Turn off the Drain Pump
	Close the Drain Valve
	Reset the Vessel to Cool
		Vessel.M101.RTS := TRUE
		Vessel.V101.RTC := TRUE
		Vessel.1.RS_COOL := TRUE

Using an SFC for Sequential Control (continued)

Naming Transitions in SFC

To name the first transition for the Ves_Ctrl SFC, follow the steps below.

1. Click on the **Select** icon on the SFC toolbar.
2. Double-click on the first transition. The Object Properties dialog box appears.
3. In the Name field of the General folder, enter `CHECK LOCKS` to rename the transition.
4. In the Comment field, enter `Check for Devices in Auto Mode`.

Setting Conditions for Transitions in SFC

To set the condition for the first transition in the Ves_Ctrl SFC, follow the steps below.

1. Select the **Condition** tab.
2. Click the **Browse** button at the bottom of the Object Properties dialog box. The Browse dialog box appears.
3. Double-click on **Vessel** in the Charts column of the CFC Charts folder. A list of the blocks in the CFC appears in the Blocks column.
4. Double-click on **M101** block. A list of I/O points for the M101 function block appears in the I/Os column.
5. Double-click on the **LOCKD** input. The following text appears in the Condition folder of the Object Properties dialog box.

   ```
   Vessel.M101.LOCKD
   ```
6. In the row with the above text, click on the down arrow in the middle of the row. A list of operators appears.
7. Click on the **=** sign to select it.
8. In the column to the right of the = sign, enter `1`, and click the cursor in the next row down. The word TRUE appears in the assignment field.
9. Repeat steps 2–8 for the **LOCKD** inputs of V101, V102, and M102.

10. Click the **Browse** button at the bottom of the dialog box again.

11. Click the **Symbols** tab. The symbol `auto_cycleYC` that you configured on page 3-25 appears in the list.

12. Double-click on the symbol name. The symbol is added to the Condition folder of the Object Properties dialog box.

13. Click on the **=** sign to select it.

14. In the column to the right of the = sign, enter `1`, and click the cursor in the next row down. The word TRUE appears in the assignment field.

15. Click on the logic box to the right of the assignment field until the word AND appears.

16. Click the **Apply** button to save the information. When the control program runs, the transition conditions must be met before the next step executes.

Using an SFC for Sequential Control (continued)

Navigating between Transitions in SFC

To configure the other transitions in the Ves_Ctrl SFC, follow the steps below.

1. Use the down-arrow button at the bottom of the Object Properties dialog box to move to the next SFC transition.

2. Repeat the procedures for naming transitions and setting conditions described above with the information in Table 3-4. You configure the CONTINUE transition on page 3-33.

Table 3-4 Transition Conditions for Ves_Ctrl

Transition Name	Comment	Condition[1]

CHECK LOCKS	Check for Devices in Auto Mode	Vessel.V101.LOCKD = TRUE
		Vessel.M101.LOCKD = TRUE
		Vessel.V102.LOCKD = TRUE
		Vessel.M102.LOCKD = TRUE
		`auto_cycleYC` = TRUE

LEVEL CHECK	Check for Level of 95%	Vessel.LI101.PV >= 95.0
AUTO CHECK	Check for TIC101 in Auto Mode	Vessel.TIC101.SMODE = 4
TEMP CHECK	Check Vessel Temp	Vessel.TIC101.PV >= 98.0

CHECK DRAIN	Confirm Drain valve is open and Drain Pump is running	Vessel.V101.OPND = TRUE
		Vessel.M101.RUNNG = TRUE

CHECK EMPTY	Check for Level < 5.0%	Vessel.LI101.PV < 5.0
STOP CYCLE	Stop auto cycle	`auto_cycleYC` = FALSE

1 For conditions with more than one expression, the first logical box must be the AND operator.

Cycling the Process

To restart the process after the fill-heat-drain sequence has executed, you must create a branch from a point after the STOP DRAIN step to a point above the FILL step. Follow the steps below:

1. In the SFC editor, select **Insert loop** icon on the SFC vertical toolbar.
2. Move the crosshair cursor to a point below the STOP DRAIN step where a horizontal line appears.
3. Click and drag until the line points above the FILL step. Release the cursor. A line with a transition box appears between the two points.
4. Configure the CONTINUE transition as follows: In the Comment field, type `Continue auto cycle`. In the Condition folder, enter `auto_cycleYC = TRUE`. The fill-heat-drain process now executes until the condition in the transition is no longer true.

Compiling and Downloading the SFC

Before you create a second SFC to start and stop the Ves_Ctrl SFC, you should compile and download the first one. Follow the steps below.

1. Select **Chart->Compile** from the menu bar of the SFC editor. The Compiling dialog box appears.
2. Select **Complete Program** and click **OK**. The program compiles. This process may take several seconds.
3. If errors occur, fix them and recompile the program. Otherwise, click **OK**.
4. Select **PLC->Download** from the menu bar of the SFC editor. The program downloads to the controller.
5. Debug the program until your process requirements are met.

Calling a Sub-SFC from the Main SFC

In order to start the Ves_Ctrl SFC, you must create another SFC to enable it. Follow these steps:

1. Click on the **Charts** container.
2. Click the right mouse button on **Charts** and select **Insert New Object->SFC** from the menu. This creates an empty SFC chart.
3. Rename the new SFC by typing `Main` over the **SFC1** name that appears highlighted in a box.
4. Double-click on the **Main** icon to open it in the SFC editor. A START step and an END step appear with a transition between them.
5. Double-click on the START step. The Object Properties dialog box appears.
6. In the Chart Comment field, enter `Start Ves_Ctrl`.
7. Select the **Processing** tab. The code that you enter here executes cyclically.
8. Click the **Browse** button at the bottom of the dialog box. The Browse dialog box appears.
9. Click the **SFC Charts** tab. The list of SFCs in the program appears.
10. Double-click on the **Ves_Ctrl** SFC. The **EN** attribute is displayed in the Attributes column.
11. Double-click on the **EN** attribute. The following text appears in the Processing folder of the Object Properties dialog box.

    ```
    Ves_Ctrl.EN
    ```
12. In the column to the right of the = sign, enter `1`, and click **Apply**. The word ON appears in the assignment field.
13. Click **Close** to close the dialog box.

Configuring the Transition for the Main SFC

To configure the transition for the Main SFC that keeps the Ves_Ctrl SFC running, follow these steps:

1. Double-click on the transition to open the Object Properties dialog box.
2. In the Name field of the General folder, enter `ALWAYS FALSE` to rename the transition.
3. Select the **Condition** tab.
4. Enter `M0.2` in the first column.
5. Click on the **=** sign to select it. A % sign appears in front of the `M0.2`.
6. In the column to the right of the = sign, enter `1`, and click **Apply**. The word TRUE appears in the assignment field. This condition should never be met; therefore, the Ves_Ctrl SFC is always activated.
7. Click **Close** to close the dialog box.

Configuring an SFC for Execution on Startup

To configure the Main SFC to start automatically when the controller goes to RUN mode, follow these steps:

1. Select **Chart->Properties** from the SFC menu bar to display the Properties SFC dialog box.
2. Select the **Run-Time Properties** tab.
3. Click on the **Search for** button to find where the Main SFC is attached to OB35.
4. When the Main SFC attached to OB35 is highlighted, click on the **Properties** button in the right of the dialog box. The Execution Main OB35 dialog box appears.
5. In the Chart field, click **On restart** to check the box.
6. Click **OK**, and click **OK** again in the Properties SFC dialog box. The Main SFC executes automatically when the controller goes to run mode.

Compile and download the program following the procedure on page 3-33. To add an OSx station and translate the function blocks and symbols to the OSx database see Chapter 2.

Chapter 4

User-defined Tag Types

4.1 Overview

Reasons for Creating a New Tag Type

If you have a device that does not map effectively into any of the standard OSx tag types, you may want to create your own user-defined tag type. A custom tag type can centralize data input and output for the device, and you can also create a tag detail or faceplate for more effective operator monitoring and control. The method of creating a new tag type described in this chapter is only valid for S7 controllers. You must be using the Engineering Toolset with the OSx Library of function blocks.

When you create a new tag type, you configure the parameters (inputs and outputs) of an S7 function block to be translated to OSx as tag attributes. You can create up to 50 new tag types. You are limited to a total of approximately 1000 attributes distributed among all user-defined tag types.

This chapter describes how to create a new tag type, by creating a tag type named TANK as an example. The SCL source file for this tag type is listed in Appendix C.

Creating a New Tag Type

To create a new tag type in OSx for S7 controllers, you must complete the following tasks:

- Create a user library (Section 4.2)
- Configure the function block name in the Symbol Table (Section 4.3)
- Create the SCL source file (Section 4.4)
- Define I/O parameters (Section 4.5)
- Configure OSx alarms (Section 4.6)
- Configure RBE alarm messaging (Section 4.7)
- Use the block in a CFC (Section 4.8)

Installing a New Tag Type in OSx

To install the new tag type in OSx, you must complete the following tasks:

- Transfer the tag from the S7 project (Sections 2.5, 2.6, and 2.7)
- Install the new tag type in OSx (Section 4.10)
- Install tags of the new tag type (Section 2.8)
- Create display faceplates for the new tag type (Section 4.13)

4.2 Creating a User Library

The most convenient way to add a new tag type is to create a User Library that is available to all S7 programs. To create a new library, follow the steps below:

1. Select **File->New** from the SIMATIC Manager menu.
2. Click on **New Library** and type the name that you want to call the library in the Name field. Then click **OK**.
3. Highlight the name of the new library and right click to display the menu selection.
4. Select **Insert New Object->S7 Program** from the menu.
5. Click on the plus signs to the left of the new library folder and the S7 Program folder to expand.
6. Highlight the Source Files and right click to display the menu selection.
7. Select **Insert New Object->External Source File** if you have developed your SCL program outside of this library or **Insert New Object->SCL Source File** if you want to start developing here.

4.3 Configuring the Tag in the Symbol Table

Naming the Tag Type

You must select an available number for the new function block. For the OSx Library and other Engineering Toolset libraries, the reserved numbers are in the range of 60 to 930. Select a number outside that range.

Enter the number and the name of the function block in the Symbol Table for the program. Give a unique, meaningful name to the new tag type. The name can be a maximum of four characters, and can contain only alphanumeric characters and the underscore. Do not use the underscore as the first character in the name, as the underscore prefix indicates system tags in OSx. Letter characters must be upper case (A–Z).

If you give the function block a name longer than four characters in the Engineering Toolset, you must use the `RENAME` keyword in the Comment field of the Symbol Table to assign it a four-character name for OSx.

Follow the steps below.

1. To access the symbol table, highlight the S7 program in the SIMATIC Manager and double-click on the **Symbols** icon that appears in the right-hand window. The Symbol Table is displayed.

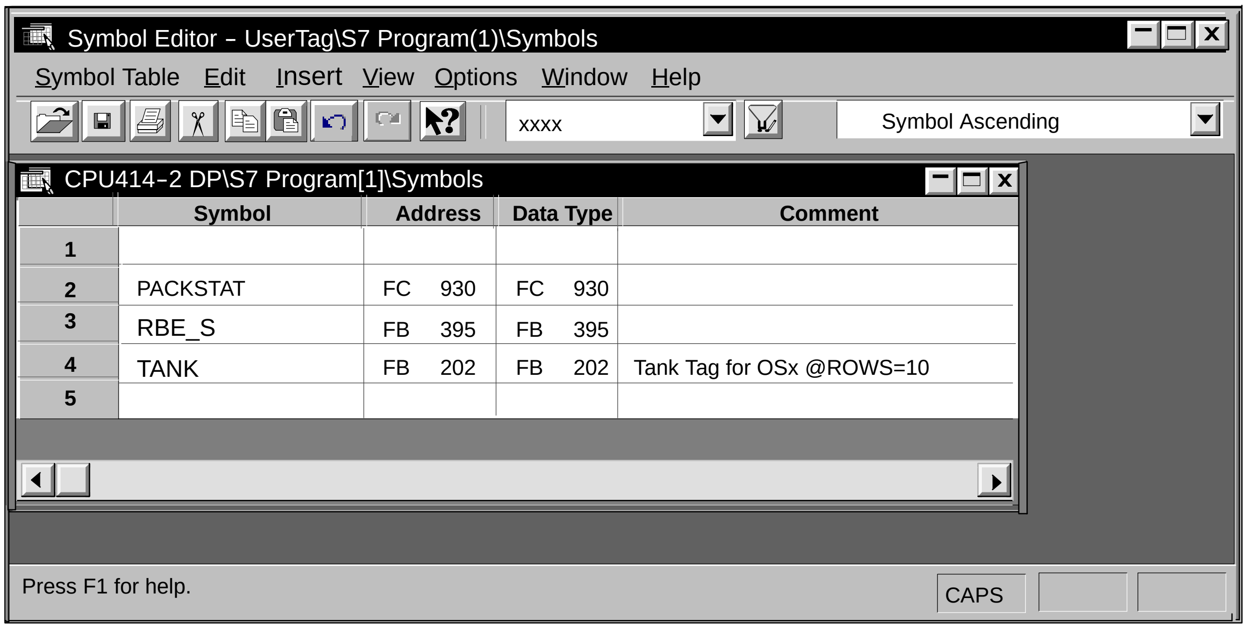
2. Enter the name that you gave the tag type (TANK, in this example) in the first blank field of the Symbol column (Figure 4-1).

3. Enter the function block number that you chose for the tag type in the Address column.

4. Enter a meaningful description in the Comment column. If needed, enter the options listed on pages 4-6 and 4-7.

5. Select **Symbol Table->Save**, and exit the Symbol Table editor.

NOTE: When you move a function block into the Blocks folder of a project or library, the block appears in the Symbol Table. For the TANK example, RBE_S and PACKSTAT have been called in the SCL code and placed in the Blocks folder of the User Tag Library. Therefore, they appear in the Symbol Table in Figure 4-1.


Symbol Editor – UserTag\S7 Program(1)\Symbols
Symbol Table Edit Insert View Options Window Help
xxxx
Symbol Ascending
CPU414–2 DP\S7 Program[1]\Symbols

	Symbol	Address	Data Type	Comment
1				
2	PACKSTAT	FC 930	FC 930	
3	RBE_S	FB 395	FB 395	
4	TANK	FB 202	FB 202	Tank Tag for OSx @ROWS=10
5				

Figure 4-1 Adding a User-defined Tag to the Symbol Table

Configuring the Tag in the Symbol Table (continued)

Comments in the Symbol Table

The options that can be assigned in the Comment field of the Symbol Table for a function block are described in the paragraphs below. After you enter the description for the function block, use the @ sign to indicate the end of the description and the beginning of the option(s).

If you enter more than one option in a Comment field, the options must be separated by at least a single space. The option assignments themselves can contain no spaces except for those involving string values enclosed in single quote marks or backslashes.

Renaming a Function Block for OSx To install a new tag type in OSx, the tag type name is limited to four characters. If a function block name is longer than 4 characters, you can give it an OSx four-character tag name, by entering the following statement in the Comment field of the Symbol Table:

```
RENAME=<new_tag_type_name>
```

where `new_tag_type_name` is a four-character name for the new tag type. The function block retains its original name in the Engineering Toolset. For example, for a function block named `BASE_STOCK_TANK`, you can enter `RENAME=TANK`.

NOTE: When you choose function block names, do not use the names of relations in the OSx database. To view a list of relation names, create a report format with the following cell function:

```
#readlist(relation:name;all;relation:name ASC)
```

Preview the report to see the list of relation names. See the *SIMATIC PCS 7 OSx Reports Manual* for guidance in creating and previewing reports.

Mapping to an Existing Tag Type To map a custom function block to an existing tag type, enter the following statement in the Comment field of the Symbol Table:

```
TAG_TYPE=<existing_tag_type_name>
```

where `existing_tag_type_name` is the name of a standard OSx tag type to which you want to map the tag. See Section 4.9 for the requirements for mapping to an existing tag type.

NOTE: The keywords `RENAME` and `TAG_TYPE` are mutually exclusive. If you use one, you cannot use the other.

Selecting Initial Number of Tags To select the initial number of instances of the tag type (up to 500) that can be installed in OSx, enter the following statement in the Comment field of the Symbol Table for the function block:

```
ROWS=n
```

where `n` is the number of tags. If no initial value is configured, `ROWS` defaults to 1. Specify the expected number of tags for this type. The number can be increased or decreased using the tag capacity utility in OSx after the tag type is installed.

Choosing a Controller Type To select a controller type (S7-414, S7-416, or S7-417), you must enter the following statement in the Comment field of the Symbol Table for the function block:

```
PLC=<pc_type>[,<pc_type>]
```

If you do not specify a controller type, the selection defaults to all S7-type controllers known to the OSx database (S7-414, S7-416, and S7-417).

4.4 Creating the SCL Source File

You create the source file for the new tag type in the SCL language. A sample SCL source code for the TANK tag is shown in Appendix C. To open the file in which you will create the source code for the function block that controls the device, follow these steps:

1. In the SIMATIC Manager, select **Source Files** under the library folder.
2. Right mouse click and select **Insert New Object->SCL Source File**. A new icon labeled **SCL Source File(1)** appears in the Source Files folder.
3. Rename the new source file by typing your function block name (for example, `TANK`) over the **SCL Source File(1)** name that appears highlighted in a box.
4. Double-click on the name of the source file to open it.
5. Select **Insert->Block Template->FB** from the menu bar. A block template for the function block appears in the source file.
6. Enter the symbol name for the function block in the first line of the template that you are creating; for example, to associate the name with a function block number in the Symbol Table, enter the following:

   ```
   FUNCTION_BLOCK “TANK”
   ```

7. Enter the name for the new tag type in the template; for example, to name the new tag type TANK, enter the following:

   ```
   NAME:TANK;
   ```

   The name TANK will be used as the OSx name for the tag type.

8. Select **Options->Customize** from SCL Editor. The Customize dialog box appears in Appendix C.
9. In the Create Block folder, click on the box to the left of **Include System Attribute ’S7_server’**. This allows the system attribute “S7_server” to be declared for an input parameter, and ensures that the function block compiles correctly.
10. If you want to use the **ENO** output, click the **OK Flag** in the Compiler folder.
11. In the Editor folder, select **Save Before Compiling** to ensure that the saved and compiled versions of the function block are the same.

4.5 Defining Function Block Parameters

Data Types for Parameters

For every parameter that you want to send to the OSx database, you must specify `S7_m_c := 'true'` for the parameter.

The following parameter data types are supported by SCL and CFC and can be used for OSx tag attributes: BOOL, BYTE, CHAR, DINT, DWORD, INT, REAL, STRING, and WORD. The following data types are also supported, but are not recommended as OSx tag attributes: TIME, BLOCK_FB, BLOCK_FC, and BLOCK_DB. Figure 4-2 shows how to specify the data type in SCL code.

```
VAR_IN_OUT
    GAUGE_LEV{ S7_m_c  := 'true'} :REAL;  // tank level
END_VAR
```

Figure 4-2 Specifying Data Type for a Parameter

Comments in SCL

The comments that you add in the SCL code for each parameter are used for OSx mapping purposes. To add comments to the parameters, use the `//` . Enter a meaningful comment that explains the parameter. If the parameter needs to have bits defined or is an enumerated type, then enter the "at" sign (@), followed by the necessary syntax for the options (Table 4-1).

If you enter more than one option in a Comment field, the options must be separated by at least a single space. The option assignments themselves can contain no spaces except for those involving string values enclosed in single quote marks or backslashes.

Defining Function Block Parameters (continued)

Table 4-1 shows the options that can be assigned in the SCL Comment field for function block parameters. These options are applicable only for the parameters using the `S7_m_c='true'` setting. See page 4-12.

Table 4-1 Options for Function Block Parameters

Option	Syntax	Task	Reference
Upload	`UPLOAD`	Mark attribute for upload.	Page 4-10
Non-networked	`NONETWORK`	Mark attribute as non-networked.	Page 4-10
Read-only	`READONLY`	Mark attribute as read-only.	Page 4-11
Not for OSx	`NOTOSX`	Do not map attribute into OSx.	Page 4-11
Deadband	`DEADBAND`	Apply deadband to the attribute. The percent of the deadband is specified in the CHANGE parameter. H_RANGE and L_RANGE specify the upper and lower limits for values for the attribute.	Page 4-11
Attribute name	`ATTR=<attribute_name>`	Specify a name different from the parameter name to be used as the attribute name.	Page 4-11
STATUS bits	`\bit_name\B<bit_number>`	Define names for bits in STATUS and STATUS-like attributes.	Page 4-14
Enumerated type	`\<name>\<value>`	Assign a name and initial value for an enumerated type attribute.	Page 4-20
	`ENUM_TYPE= <enum_type_name>`	Specify enumerated type definition.	
Alarms	`ALARM=<alarm_bits>`	Specify alarm bits for attribute.	Page 4-22
Alarm limits	`LIMIT=<alarm_bit> [,<criticality>]`	Specify limit value association and alarm priority for alarm bit(s).	Page 4-22

Configuring Upload To improve data collection performance, you can enter `UPLOAD` for ranges and alarm limit values, which rarely or never change. This causes OSx to read the tag attribute at a slower rate, every other event scan period, compared to every event scan period for attributes without `UPLOAD`.

Configuring Attributes as Non-Networked Enter `NONETWORK` in the Comment field of an attribute if you want the value to reside in the OSx database but not in the controller program. For example, in an IVAR or CALC tag, high and low range values are often non-networked values that can be scaled and used in OSx in bar graph and meter DDOs. Other examples of attributes that could be non-networked are **CHANGE** and **UNITS** of measure.

Configuring Attributes as Read-Only Enter `READONLY` in the Comment field of an attribute if you want the attribute to be read-only in OSx. The **STATUS** attribute is normally configured as read-only.

Configuring Attributes as Not OSx The `NOTOSX` keyword allows you to install a pre-existing function block as a tag type in OSx. The pre-existing block could have parameters that are not needed or not applicable for OSx usage. If so, enter `NOTOSX` in the Comment field of the parameter and the parameter will not be mapped to OSx.

Configuring Attributes as Deadband **H_RANGE** and **L_RANGE** are reserved attributes that OSx uses to represent the upper and lower limits for values for that tag. These limits apply to attributes that you designate as having a `DEADBAND` in the Comment field assignment.

In OSx, `DEADBAND` means that a deadband is applied to the parameter. The deadband percentage is specified with the **CHANGE** attribute, which can be configured as a parameter in the SCL code or specified in the Comment field. OSx uses the deadband to minimize database updates due to small changes in the value of a parameter. If all value changes are significant, set the **CHANGE** attribute to a value of 0.0. Otherwise, set the **CHANGE** attribute to a percentage of the engineering range. See page 2-11 for details.

The data types of **H_RANGE**, **L_RANGE**, and the attribute to which the deadband is applied must be identical. For example, if you want to apply a deadband to the parameter named **GAUGE_LEV**, which is a real value, then **H_RANGE** and **L_RANGE** must also be real values.

Only one **H_RANGE** and one **L_RANGE** can be configured for a tag. For example, if you have two different measurements, such as level and temperature in a tank tag, you can specify the ranges for one or the other, but not both. The level may range from 0 to 50 feet while the temperature range is 0 to 320 degrees Celsius. See Figure 4-12 on page 4-24 for the TANK example.

Configuring the Attribute Name The `ATTR` keyword allows you to specify the name of the attribute in the OSx database table that represents this parameter. This option can be used for new tag types as well as for function blocks that are to be mapped to existing OSx tag types.

Defining Function Block Parameters (continued)

Using the New Tag Type in Trends

In order to trend analog values from a new tag type in OSx, the following conditions must be met:

- The attributes **H_RANGE** and **L_RANGE** must be defined.
- At least one attribute must have a `DEADBAND`

If these conditions are met, all attributes of the same data type as the **H_RANGE** and **L_RANGE** can be used for trending.

In order to trend digital values from a new tag type in OSx, the parameters must be declared as `ENUM_TYPE`.

Defining S7 System Attributes to Parameters

Table 4-2 lists the system attributes for the SCL compiler. Use these attributes when you define the I/O for your function block. For example, if you want the parameter to be displayed on the block in the CFC editor, type `S7_visible := 'true'` in braces after the name of the I/O parameter. Any parameter that is to be an attribute in the OSx station must have `S7_m_c := 'true'` in braces. The Tank SCL code in Appendix C shows examples of using the S7 system parameters.

Table 4-2 S7 System Attributes for Parameters

Attribute	Value	Assign This Attribute When	Permitted Declaration Type
S7_server	alarm_archiv	The parameter is relevant for message configuration. This parameter contains the message number.	IN, only for parameter EV_ID
S7_a_type	alarm_8p, alarm_s	You specify the block type of the message block that will be called in the declaration section. The attribute S7_server := alarm_archiv must also be assigned.	IN, only for parameter EV_ID
S7_m_c	true, false	The parameter can be monitored or changed from an operator panel.	IN/OUT/IN_OUT
S7_visible	true, false	The parameter will or will not be displayed in the CFC.	IN/OUT/IN_OUT
S7_link	true, false	The parameter will or will not be linked in the CFC.	IN/OUT/IN_OUT
S7_param	true, false	The parameter will or will not be assigned values in the CFC.	IN/IN_OUT

Defining the IN_OUT Parameters

Data that can be monitored and changed by an operator must be defined as an in/out parameter. In the source code template, you specify in/out parameters to pass data to the called function block, process data there, and store the results sent from the called block in the same variables again.

For every parameter that you want to send to the OSx database, you must specify `S7_m_c := 'true'`. For example, if the operator needs to change the low-low alarm for the TANK function block during the process, configure the low-low alarm parameter as an IN_OUT variable. Figure 4-3 shows how to define a parameter so that the operator can modify the value.

```
VAR_IN_OUT

  LL_ALM   { S7_m_c   := 'true';  S7_visible := 'false'}
        :REAL;   //@LIMIT=0x0100,CRIT

END_VAR
```

Figure 4-3   Specifying IN_OUT Parameters

The IN_OUT parameters cannot be assigned an initial value in SCL. To assign an initial value to these parameters, compile the SCL and follow the steps below.

1. Select **SIMATIC->STEP7->LAD, STL, FBD – Programming S7 Blocks** editor.
2. Select **File->Open...**. In the Entry Point field of the Open dialog box, select **Library**. In the Name field select the library that contains the new tag type.
3. In the Block folder of the library, click the function block to open it in the STL editor.
4. Assign the value in the Initial Value column for the parameter.
5. Save and close the object. If you recompile the object in SCL after assigning the initial values, repeat the steps above to load the initial values again.

Defining the OUTPUT Parameters

In the source code template, you specify output parameters that send data processed by the block to other function blocks or field devices. You can use these outputs to connect to other function blocks or I/O in the CFC editor. A sample listing for the function block TANK is shown in Figure 4-4:

```
VAR_OUTPUT

     INHHA     :BOOL:=0;// Level in high-high alarm (GAUGE_LEV > HHA)
     INHA      :BOOL:=0;// Level in high alarm (GAUGE_LEV > HA)
     INLA      :BOOL:=0;// Level in low alarm (GAUGE_LEV < LA)
     INLLA     :BOOL:=0;// Level in low-low alarm (GAUGE_LEV < LLA)

END_VAR
```

Figure 4-4 Specifying Output Parameters

Defining the STATUS Attribute

All OSx tags have a **STATUS** attribute. If the function block does not have a **STATUS** parameter, then one is created automatically in OSx. If you define a **STATUS** parameter, there are three different ways to define the individual bits.

- Use an existing status word for an OSx tag.
- Define the bits in the **STATUS** parameter's Comment field.
- Define the bits in files on the OSx station.

The **STATUS** parameter must be declared as a WORD, where B1 is the most significant bit and B16 is the least significant bit. The three least significant bits (B14, B15, and B16) are reserved for OSx specific processing. The other thirteen bits can be used for customizing the information sent to OSx.

Using an Existing OSx Status Word

If the status of the user-defined tag type can be mapped as an existing OSx tag type, enter `ENUM_TYPE=<status table name>` in the STATUS parameter Comment field, where `status table name` is the name of the status table for the OSx tag type. Table 4-3 shows the status table name for each OSx tag type. To view how the status bits are mapped in OSx, see the chapter on Defining Tags in the *SIMATIC PCS 7 OSx Process Configuration Manual*.

NOTE: If you want to trend only the most significant bit of a STATUS word, the most efficient way to define the STATUS parameter is to enter `ENUM_TYPE=di_qual` in the STATUS parameter Comment field.

Table 4-3 Status Table Names for OSx Tags

OSx Tag Name	Status Table Name
AI	ai_qual
AO	ao_qual
DI	di_qual
DO	do_qual
LOOP	loopqual
TMR	tmr_qual
CTR	ctr_qual
CALC	dataqual
IVAR	dataqual
DEVICES (motors, valves, presses, cylinders)	dev_qual
DI10	di10qual
DO10	do10qual
TEXT	tx_qual

Defining a Status Word in the Comment Field

The syntax for defining bits is `\CISTRING10\Bnn`, where `CISTRING10` is a case-insensitive bit name of up to 10 characters and `nn` is the bit number defined in the string. Use B0 to define a name for the zero value (when no bit is set).

If the most significant bit of the status word indicates that a motor is running, you would type `\RUNNING\B1` after the @ sign in the Comment field for the **STATUS** parameter. Choose names that represent meaningful indications for the status text used in OSx. These status bit names can correspond to the SCL variable names that you use in the source code.

Remember to put a space between bit assignments. In the project, the comments are displayed in the Comment field of the function block if you open it in the Blocks folder or in the Block Object Properties I/O folder of the CFC editor.

If these definitions take less than 80 characters, the **STATUS** parameter's Comment field has the format shown below.

```
//@\HH_ALM\B5 \H_ALM\B6 \L_ALM\B7 \LL_ALM\B8 \L_TEMP\B9
\H_TEMP\B10 \HH_TEMP\B11 \LL_TEMP\B12
```

Defining a Status Word in Files on OSx

If the definition is more than 80 characters, you must include it in a file in the /**usr/tistar/data/db/tables/local/enum_types** directory on OSx. The file name consists of up to 12 lower-case characters and the extension **.ddl**. When you define the **.ddl** file for a **STATUS** attribute, you must specify the enumerated type “`AS STATUS OF`” in the first line.

For the sample definition shown in Figure 4-5, you name the file **tkqual.ddl**. To use this definition for the **STATUS** attribute, type the assignment `ENUM_TYPE=tkqual` in the Comment field of the **STATUS** parameter.

```
DEFINE tkqual AS STATUS OF
(HH_ALM    =    0x0800,
H_ALM      =    0x0400,
L_ALM      =    0x0200,
LL_ALM     =    0x0100,
L_TEMP     =    0x0080,
H_TEMP     =    0x0040,
HH_TEMP    =    0x0020,
LL_TEMP    =    0x0010);
```

Figure 4-5 Defining the STATUS Attribute in a File

You can set the **STATUS** word value in the SCL code by using the PACKSTAT function. Figure 4-6 shows an example for the TANK tag. The actual bit names correspond to the Boolean I/O parameters of the block that you create or Boolean variables used in the SCL code. The PACKSTAT function works only if the **STATUS** parameter is defined as a WORD.

```
STATUS := PACKSTAT (B1:=0, B2:=0,
          B3     :=     0,
          B4     :=     0,
          B5     :=     HH_ALM,
          B6     :=     H_ALM,
          B7     :=     L_ALM,
          B8     :=     LL_ALM,
          B9     :=     L_TEMP,
          B10    :=     H_TEMP,
          B11    :=     HH_TEMP,
          B12    :=     LL_TEMP,
          B13    :=     0,
          B14    =      0,
          B15    :=     0 );
```

Figure 4-6 Packing Bits for the STATUS Attribute

Defining Function Block Parameters (continued)

Defining Other Bit-Oriented Attributes

In addition to the **STATUS** attribute, you may configure other bit-oriented attributes to monitor specific states or conditions in the device. If you have other bit-oriented attributes for which you need to define individual bits, you must declare the parameters as data type WORD.

The syntax for defining bits is `\CISTRING10\Bnn`, where `CISTRING10` is a case-insensitive bit name of up to 10 characters and `nn` is the bit number defined in the string. Use B0 to define a name for the zero value (when no bit is set). The definitions of the bits must fit within the parameter Comment field, which is sized at 80 characters. Remember to put a space between bit assignments.

If these definitions take less than 80 characters, the **TANK_STS** parameter's Comment field has the following format:

```
//@\OOS\B1 \MANUAL\B2 \RUN\B3 \FILL\B4 \EMPTY\B5
\CLOSE\B6 \RAISE\B7 \HALT\B8
```

If the definition is more than 80 characters, you must include it in a file in the **/usr/tistar/data/db/tables/local/enum_types** directory on OSx. The file name consists of up to 12 characters and the extension **.ddl**. When you define a WORD parameter (not **STATUS**) in the **.ddl** file to be used as a bit mask, you must specify the enumerated type `AS SET OF` in the first line.

NOTE: A parameter that you define as WORD uses the syntax `AS SET OF` to define the meaning of each bit (Figure 4-7). A parameter that you define as INT uses the syntax `AS` for listing the possible values in the **.ddl** file (Figure 4-9).

For the sample definition shown in Figure 4-7, the parameter is defined as a WORD and you name the file **tksts.ddl**. To use this definition for the **TANK_STS** attribute, type the assignment `ENUM_TYPE=tksts` in the Comment field of the **TANK_STS** parameter.

```
DEFINE tksts AS SET OF
(OOS           =    0x8000,
MANUAL         =    0x4000,
RUN            =    0x2000,
FILL           =    0x1000,
EMPTY          =    0x0800,
CLOSE          =    0x0400,
RAISE          =    0x0200,
HALT           =    0x0100,
LOW            =    0x0080,
SEARCHH2O      =    0x0040,
ABORTH2O       =    0x0020);
```

Figure 4-7 Defining Other Bit-Oriented WORD Parameters in a File

To put the bits in the attribute for a WORD parameter, enter the PACKSTAT function call in the SCL source code, using the syntax shown in the example below. The actual bit names will correspond to the Boolean I/O parameters of the block that you create or Boolean variables used in the SCL code. The PACKSTAT function works only for parameters defined as a WORD (Figure 4-8).

```
TANK_STS := PACKSTAT (B1 := OOS,
               B2     :=     MANUAL,
               B3     :=     RUNNING,
               B4     :=     FILLING,
               B5     :=     EMPTYING,
               B6     :=     CLOSED,
               B7     :=     RAISE,
               B8     :=     HALT,
               B9     :=     LOWER,
               B10    :=     SEARCH_H2O,
               B11    :=     ABORT_H2O
               B1     :=     0,
               B13    :=     0,
               B14    :=     0,
               B15    :=     0,
               B16    :=     0);
```

Figure 4-8 Packing Bits for the TANK_STS Attribute

Creating Enumerated-Type Parameters

Enumerated-type parameters are used to associate a value with a name; for example, 32768 = ON and 0 = OFF. The entry in the Comment field of the attribute has the following syntax:

```
\CISTRING10\V
```

where `CISTRING10` is a 10-character name for the enumerated member, and `V` is the member's numeric value. Assign enumerated parameters the data type WORD.

The definitions of the enumeration members must fit within the parameter Comment field, which is sized at 80 characters. Remember to put a space between value assignments. If these definitions take less than that number of characters, the TANK_MD parameter's Comment field has the following format:

```
//@\OOS\999 \MANUAL\10 \RUN\9 \FILL\8 \EMPTY\7 \CLOSE\6
\RAISE\5 \HALT\4 \LOWER\3 \SEARCHH2O\2 \ABORTH2O\1
\NOMODE\0
```

If the definition is more than 80 characters, you must include it in a file in the /**usr/tistar/data/db/tables/local/enum_types** directory on OSx. The file name consists of up to 12 lower-case characters and the extension **.ddl**. When you define an INT or WORD parameter in the **.ddl** file to be used as a value, you must specify the enumerated type with "AS" in the first line.

NOTE: When you use an INT or WORD value enumerated type, you must define the value zero. That is, "`\CISTRING10\0`" must be defined with your member name in place of `CISTRING10`. If you are defining the enumerated type using a **.ddl** file, then a member name must be defined for "`=0`" (see Figure 4-9).

For the sample definition shown in Figure 4-9, you name the file **tkmode.ddl**. To use this definition for the **TANK_MD** attribute, type the assignment `ENUM_TYPE=tkmode` in the Comment field of the **TANK_MD** parameter.

```
DEFINE tkmode AS (OOS=999,
MANUAL         =     10,
RUN            =     9,
FILL           =     8,
EMPTY          =     7,
CLOSE          =     6,
RAISE          =     5,
HALT           =     4,
LOWER          =     3,
SEARCHH2O      =     2,
ABORTH2O       =     1,
NOMODE         =     0);
```

Figure 4-9 Defining Enumerated Parameters in a File

4.6 Configuring OSx Alarms

In a process, certain conditions must be monitored for safety or other reasons. Different levels of monitoring are required for different stages in the process based upon the incoming field data. You create alarms to notify an operator of unsafe or critical conditions. Most alarms are based upon process values passing specified limits.

You can specify up to 10 alarms for each tag type that you create. Any parameter that generates an alarm must be assigned an alarm value. You must also assign a limit value corresponding to the bit location in the status attribute of the alarm parameter. These assignments ensure that the OSx alarming features described in the chapters on alarm configuration in the *SIMATIC PCS 7 OSx Process Configuration Manual* can be utilized.

To configure alarms, you must specify in the Comment field of the alarm parameter where the alarm bits are located in the **STATUS** word. Specify the alarms bits in hexadecimal representation. Multiple bits can be specified, but you cannot use the OSx reserved bits 0x0001, 0x0002, and 0x0004.

The ALARM keyword in the parameter Comment field specifies the function block parameter that is monitored for alarm purposes. Within the SCL code, you create an alarm algorithm to compare the value of this parameter against the values of other parameters that represent alarm limits. The value assigned to the ALARM keyword is the set of all possible bit values that can be set in the **STATUS** word when the limits are exceeded.

The LIMIT keyword specifies the function block parameter that is an alarm limit in the alarm algorithm. The value that you assign to the LIMIT keyword is the bit value that is set in the **STATUS** word when the limit is exceeded.

When you specify the status bits for alarm generation, you must also specify the limit bits for each alarm bit. For example, if you specify ALARM=0x8100 for a parameter, you must configure LIMIT=0x8000 and LIMIT=0x0100 in the Comment field for the associated limit values. If no limit bits are assigned, the tag type will not support alarm capability.

All loop alarms are generated based on the pv attribute. The limit specifies the bit in the status word that is set when the alarm value is exceeded. Figure 4-10 shows how the Comment field assignments for the PID block would be made in the Engineering Toolset for the standard OSx LOOP tag type.

Parameter/ Attribute	Comment Field Assignment
pv	ALARM=0x1fe0
hh_alm	LIMIT=0x1000
h_alm	LIMIT=0x0800
l_alm	LIMIT=0x0400
ll_alm	LIMIT=0x0200
h_dev	LIMIT=0x0100
l_dev	LIMIT=0x0080
roc_alm	LIMIT=0x0040
status *	LIMIT=0x0020

* This is for the bad transmitter bit.

Figure 4-10 Loop Alarm Configuration

Configuring OSx Alarms (continued)

For status and other bit-oriented parameters, the ALARM and LIMIT assignments can both be present in the attribute's Comment field and you can specify more than one bit in the LIMIT assignment. You can assign an alarm priority level of INFO, WARN, or CRIT to the associated alarm. If you do not specify an alarm priority, it defaults to INFO.

Figure 4-11 illustrates this usage in the case of the status attribute of the standard UNIT tag type. The UNIT example sets up alarm configuration data for the four alarms represented in the UNIT tag type's status word (ABORT [0x8000], ALARM [0x4000], HOLD [0x2000], WAIT [0x0800]) and assigns the alarm priority WARN to all the alarms. For further information on the UNIT tag, see the *SIMATIC PCS 7 OSx Batch Programming Manual*.

```
Parameter/       Comment Field
Attribute        Assignment
----------       --------------       ----------
status           ALARM=0xe800 LIMIT=0xe800, WARN
```

Figure 4-11 Unit Alarm Configuration

The TANK example assigns different alarm priorities to individual bits. The **HH_ALM** is set at 90 with a critical priority. When the **GAUGE_LEV** exceeds 90, the **HH_ALM** bit (0x0800) in the **STATUS** parameter is set within the TANK block. The high-high alarm defaults to a critical alarm when used in alarm configuration on OSx. Figure 4-12 shows the parameter Comment field assignments for the user-defined TANK tag type.

```
Parameter/       Comment Field
Attribute        Assignment
----------       --------------
gauge_lev        ALARM  =0x0f00
hh_alm           LIMIT  =0x0800,CRIT
h_alm            LIMIT  =0x0400,WARN
l_alm            LIMIT  =0x0200,WARN
ll_alm           LIMIT  =0x0100,CRIT
temp             ALARM  =0x00f0
l_temp           LIMIT  =0x0080,WARN
h_temp           LIMIT  =0x0040,WARN
hh_temp          LIMIT  =0x0020,CRIT
ll_temp          LIMIT  =0x0010,CRIT
```

Figure 4-12 Tank Alarm Configuration

Alarm Limitations for User-defined Tags

When you make the ALARM and LIMIT assignments described on page 4-22, the tags will support basic alarm functionality within OSx. However, there are limitations for user-defined tags that use custom enum_types defined in the **.ddl** file. If you use the standard enum_types listed in Table 4-3 on page 4-15, these limitations do not apply.

- Status values in alarm log messages appear in hexadecimal format rather than text format.
- Embedded values in action request summaries and log messages appear in hexadecimal format.
- Alarm-related targets are not available on tag detail displays that you create for a user-defined tag type.

NOTE: The data that translates hexadecimal tag status values into meaningful display text, such as ON or OFF, or engineering units, such as LITERS, is contained in the **stcm** and **stcm_dict** tables in the OSx database. You can make use of user-reserved entries in these tables to customize your tag type, but a detailed explanation of how to configure these entries is beyond the scope of this manual. See the section on Customizing the State Names in the Editors chapter of the *SIMATIC PCS 7 OSx Graphical Editor Manual*.

4.7 Configuring RBE Alarm Messaging

Defining the INPUT Parameters for RBE

In the source code template, you specify input parameters that receive and pass data to the function block. In the SCL code, a parameter that is defined as an input cannot be modified.

There is one required VAR_INPUT for RBE functionality: the EV_ID input. Enter the code shown in Figure 4-13 in the VAR_INPUT field:

```
VAR_INPUT

        EV_ID {S7_m_c   :=    ’true’;
        S7_visible      :=    ’false’;
        S7_link         :=    ’false’;
        S7_param        :=    ’false’;
        S7_server       :=    ’alarm_archiv’;
        S7_a_type       :=    ’alarm_s’}
        :DWORD          :=    0;  //Identity number for message

END_VAR
```

Figure 4-13 Defining INPUT for RBE

Defining the IN_OUT Parameters for RBE

Certain IN_OUT parameters are required for proper RBE functionality in OSx. Declare these IN_OUT parameters under the VAR_IN_OUT declaration field as shown in Figure 4-14. These parameters must be declared together in the order listed.

```
VAR_IN_OUT

//–––OSx interface variables
     AND_MASK{S7_m_c:=’true’;S7_visible:=’false’}: WORD;
     OR_MASK{S7_m_c:=’true’;S7_visible:=’false’}: WORD;
     STATUS{S7_m_c:=’true’;S7_visible:=’false’}:   :WORD;
     //See page 4-14 for information to use the comment field to map bits to the
     //status word and send to OSx.
     HEALTH{S7_m_c:=’true’;S7_visible:=’false’}: WORD;

END_VAR
```

Figure 4-14 Specifying IN_OUT Parameters for RBE

Calling RBE Message Blocks

If you create a block to generate alarms, you use the RBE_S or RBE_P function blocks in the OSx Library to provide an event-driven message to OSx. OSx uses two different types of messaging for the S7 controllers, ALARM_S and ALARM_8P. ALARM_S messages share a common resource pool in the controller and may be delayed waiting for the next resource during periods of heavy, sustained message activity. ALARM_8P messages have dedicated resources and are therefore processed immediately. They should be used for critical alarms.

For user-defined tag types, **STATUS** is the only attribute that is sent to OSx when an RBE message is generated. In the OSx Library, RBE_S and RBE_P function blocks act as an alarm interface between a function block and the OSx station. All OSx Library blocks use RBE_S except for the AI and LOOP blocks, which use RBE_P. A block can contain only one event message, either RBE_S or RBE_P.

In order for the controller program to call an RBE_S message block to be used with ALARM_S, you must define the variables shown in Figure 4-15.

```
VAR
//---Prior call values---
    PRIOR_STATUS      :WORD;     //Value of STATUS on last scan
    PRIOR_AND_MASK    :WORD;     //Value of AND_MASK on last scan
    PRIOR_OR_MASK     :WORD;     //Value of OR_MASK on last scan

    RBE_S_1           :RBE_S;    //RBE_S instance
END_VAR
```

Figure 4-15   Specifying RBE_S Variables

You must insert the following function call at the end of the function block code in order for RBE_S to generate and control the exception. Follow the syntax shown in Figure 4-16. You also must copy the RBE_S block from the OSx Library to the Blocks folder of the library.

```
RBE_S_1 (EV_ID    :=     EV_ID,
    STATUS        :=     STATUS,
    AND_MASK      :=     AND_MASK,
    OR_MASK       :=     OR_MASK,
    HEALTH        :=     HEALTH,
    P_STAT        :=     PRIOR_STATUS,
    P_AND         :=     PRIOR_AND_MASK,
    P_OR          :=     PRIOR_OR_MASK);
```

Figure 4-16   Calling RBE_S in SCL

Configuring RBE Alarm Messaging (continued)

In order for the controller program to call an RBE_P message block to be used with ALARM_8P, you must define the local variables shown in Figure 4-17.

```
VAR
//---Prior call values---
        PRIOR_STATUS      :WORD;        //Value of STATUS on last scan
        PRIOR_AND_MASK :WORD;        //Value of AND_MASK on last scan
        PRIOR_OR_MASK   :WORD;        //Value of OR_MASK on last scan

        RBE_P_1              :RBE_P;       //RBE_P instance
END_VAR
```

Figure 4-17   Specifying RBE_P Variables

You must insert the following function call at the end of the function block code in order for RBE_P to generate and control the exception. Follow the syntax shown in Figure 4-18. You also must copy the RBE_P block from the OSx Library to the Blocks folder of the library.

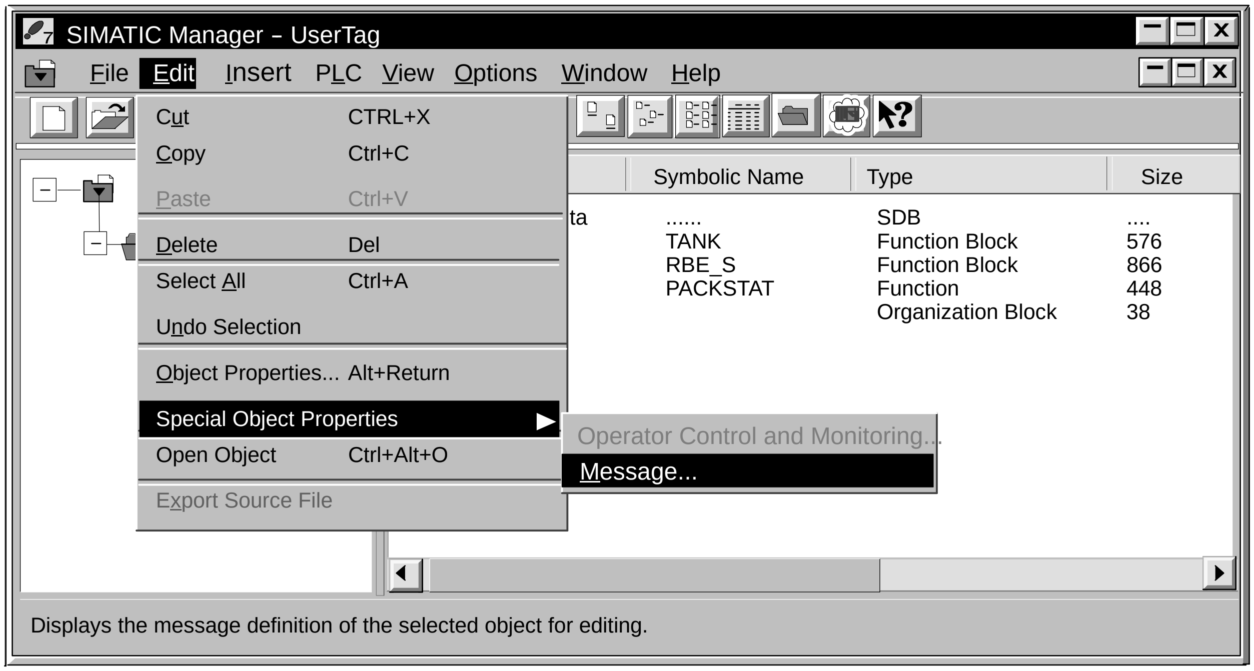
```
RBE_P_1 (EV_ID    :=      EV_ID,
     STATUS       :=      STATUS,
     AND_MASK     :=      AND_MASK,
     OR_MASK      :=      OR_MASK,
     HEALTH       :=      HEALTH,
     P_STAT       :=      PRIOR_STATUS,
     P_AND        :=      PRIOR_AND_MASK,
     P_OR         :=      PRIOR_OR_MASK);
```

Figure 4-18   Calling RBE_P in SCL

Configuring S7 Alarm Messaging

Once the SCL source file is compiled, you need to configure alarm messaging for OSx if your block has used RBE_S or RBE_P. Follow the procedure below.

1. Select the Blocks folder of the library program to display the blocks.
2. Next highlight the function block created from the SCL file. For example, FB202 for the tank tag.
3. Select **Edit->Special Object Properties->Message** from the SIMATIC Manager menu. The Message Configuration dialog box appears.



......	SDB	....
TANK	Function Block	576
RBE_S	Function Block	866
PACKSTAT	Function	448
	Organization Block	38

Displays the message definition of the selected object for editing.

Figure 4-19 Selecting Alarm Messaging

4. Click on the **New Device** button to add the OSx Display Device. The Add Display Device dialog box appears.

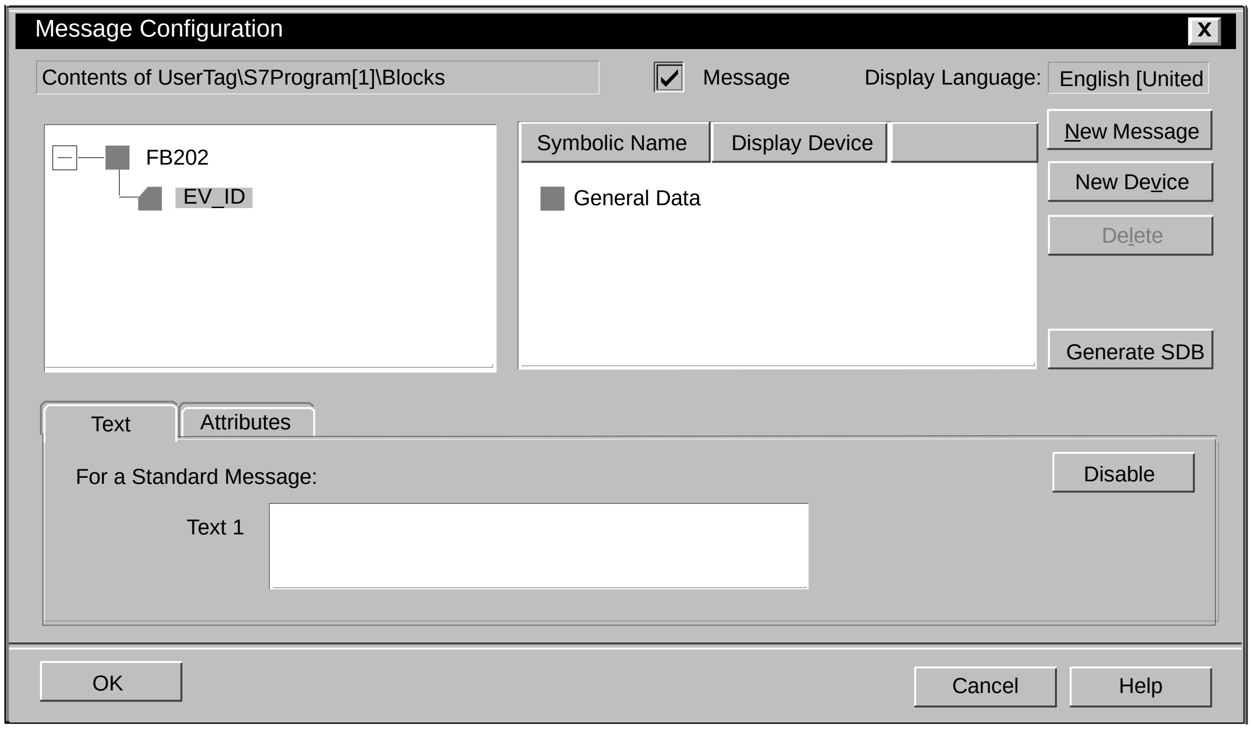
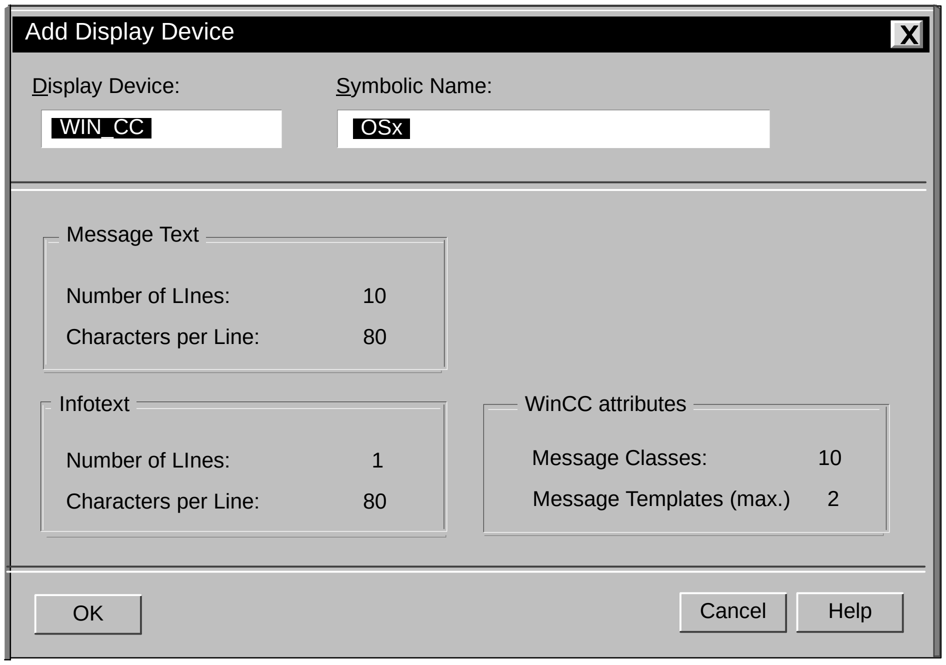


Figure 4-20 Message Configuration

5. In the Display Device field, select **WIN_CC** by using the select list.

6. In the Symbolic Name field, type `OSx` and then click **OK**.

7. In the Text folder, type $$AKZ$$ in the Text 1 field. Select the **Disable** button to lock this field. Then click **OK**.



Display Device: WIN_CC

Symbolic Name: OSx

Message Text

Number of LInes:	10
Characters per Line:	80

Infotext

Number of LInes:	1
Characters per Line:	80

WinCC attributes

Message Classes:	10
Message Templates (max.)	2

Figure 4-21 Adding OSx Device for Alarm Messaging

4.8 Using the Block in a CFC

Once the custom block has been compiled and tested, it can be used like any other function block in a CFC. From the CFC in your project, expand the library that contains the new block, and then select and drag the block into a sheet in the CFC editor. You can now configure it as described in Chapter 2.

4.9 Mapping a User-defined Function Block to an Existing Tag Type

You can create a custom function block to be mapped to an existing OSx tag type. This feature is intended for cases where the user's new tag type has the exact same set of attributes and supports the same general functionality as an existing standard or previously established user-defined tag type. The existing OSx tags are described in the chapter on defining tags in the *SIMATIC PCS 7 OSx Process Configuration Manual*.

The differences between the new and existing tag types must be completely hidden within the function block implementing the type, possibly in the form of some alteration to the program.

A good example is the collection of motors and valves in the OSx library. For each OSx motor or valve tag type, there are several function blocks that map to it.

For parameters with S7_m_c set to true, the major constraint on this type of reuse is that the parameter sets of the new and existing types must match exactly, specifically in the following ways:

- Same number of mapped parameters/attributes.
- All corresponding parameter/attributes have identical names.
- All corresponding parameter/attributes are of the same data type.

To map a user-defined function block to an existing OSx tag type, enter the following assignment in the Comment field of the Symbol Table for the function block type:

```
TAG_TYPE=<existing_tag_type_name>
```

where `<existing_tag_type_name>` is the name of a standard OSx tag type.

4.10 Installing a New Tag Type

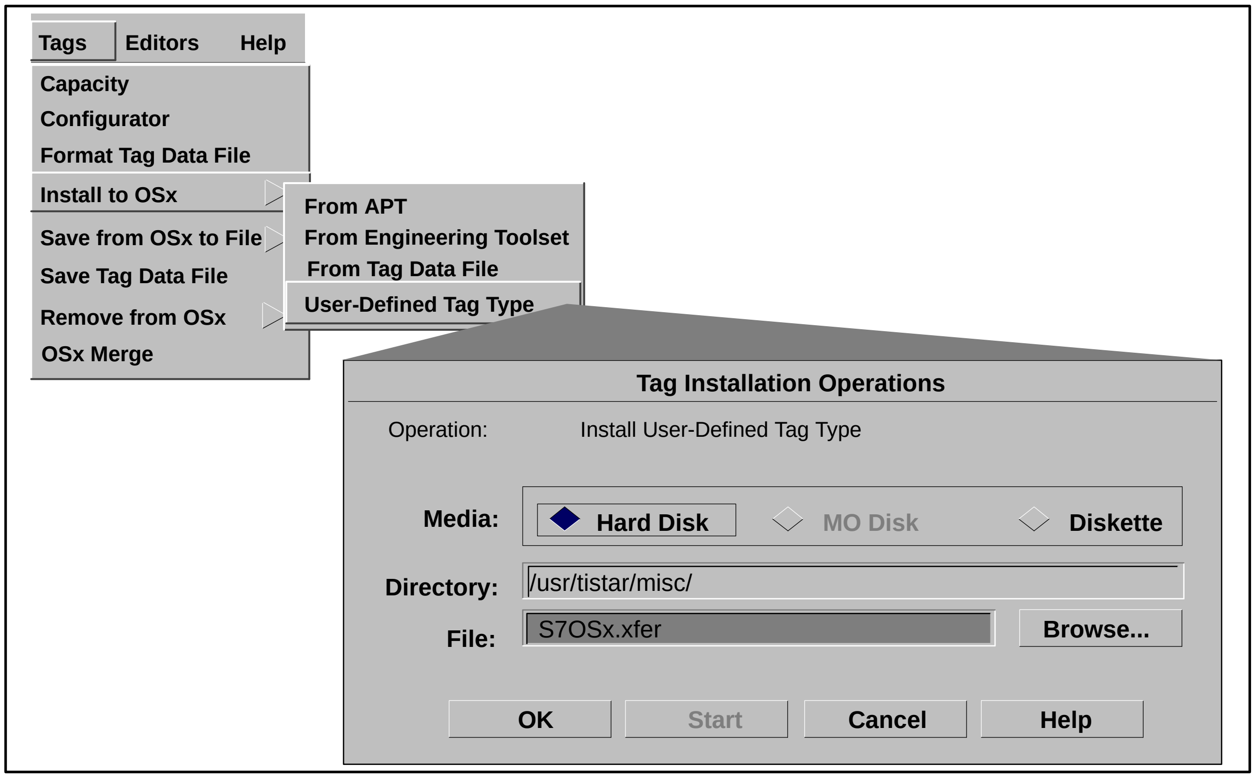
You can create your own custom-designed tag types based on function blocks defined in the S7 Engineering Toolset environment. Once you have installed these user-defined tag types in OSx, you can create instances of the tag that will be recognized and processed by OSx.

In a multiple-station system, you must assign the sysadmin role to the OSx station where you want to install the new tag type. See the *SIMATIC PCS 7 OSx System Administration Manual* for details.

To install a new tag type that you developed in the S7 Engineering Toolset, follow the steps below:

1. Select **Controls->Change System State** from the main menu bar. Select **Offline** to set OSx to the Offline state.

2. From the main screen of OSx, select **Tags->Install to OSx->User-Defined Tag Type**. The Tag Installation Operations dialog box appears (Figure 4-22).



Operation: Install User-Defined Tag Type

Media: Hard Disk

Directory: /usr/tistar/misc/

File: S7OSx.xfer

Figure 4-22 Installing New Tag Types

3. If you are installing the new tag type from a diskette, click the button to the left of **Diskette**.

   If you are installing the new tag type from the **S7OSx.xfr** in the **/usr/tistar/misc** directory on your hard drive, click the button to the left of **Hard Disk**.

4. Use the **Browse** button to select the tag type that you want to install.

5. Click the **OK** buttons on the Select List and on the Tag Installation dialog box to begin installing the new tag type. OSx begins resizing the database. During the Resize state, OSx logs you off. When resizing is complete, the station changes to the Offline state.

6. Log on again to continue.

7. On a multiple-station system, assign the primary role to the sysadmin station in order to propagate the new tag type to the other OSx stations. Resynchronization of all OSx stations is required.

NOTE: After you install a new tag type, any custom programs that reference constants in **.t** and **.r** header files related to the new tag type, such as custom BCL programs or custom reports, must be compiled. The **.t** and **.r** header files hold database definitions of the new tag type and are located in the **/usr/tistar/include/enum_types** directory.

4.11 Modifying a User-defined Tag Type

In order to modify a user-defined tag type, you must first delete all tags of that type in OSx. You must then delete the tag type itself. See Section 4.12 for guidelines and a procedure.

Make the necessary modifications to the function block in the Engineering Toolset, and reinstall the tag type as described in Section 4.10.

4.12 Deleting a User-defined Tag Type

Before you can delete a user-defined tag type, you must first remove all the instances of that tag type in your system. To remove tags individually, select **Tags->Remove from OSx->By Tag Name** from the main menu bar. The tag selection list provides filtering based on tag type, making it easier to locate the affected tags.

Tags that are being used in certain OSx processes must be removed from those processes before they can be deleted. The following cases require special handling:

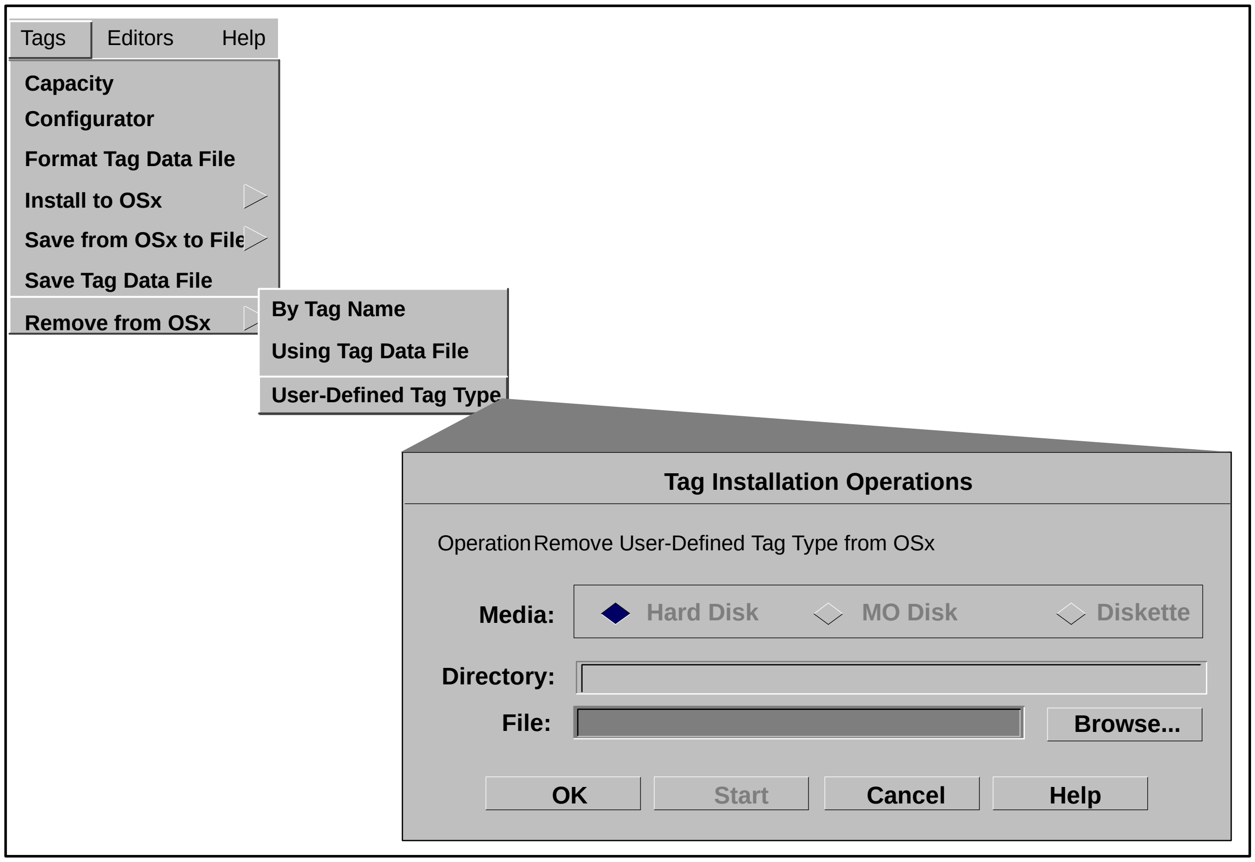
- The tag is an alarm group state tag. You must first remove the tag from the alarm group in the Alarm Group Configuration dialog box. Then you can delete the tag from the system.
- The tag is a component of a recipe area that currently has a production recipe. To delete the tag, you must first remove its recipe from production. See the *SIMATIC PCS 7 OSx Recipe Manual* for the procedure.
- The tag is a recipe area tag that currently has a production recipe. To delete the tag, you must first remove the recipe in that area from production. See the *SIMATIC PCS 7 OSx Recipe Manual* for the procedure.
- The tag is configured in an action request. This includes trigger, reset, answer, or embedded tags. You must first either delete the action request or replace the tag in the action request with another tag. In the case of a nonessential embedded tag, you can simply remove the tag from the action request. Then you can delete the tag from the system.
- The tag is used in a report, graphic, BCL, or RDT program. You must remove the tag from these items manually. Then you must recompile (for programs), revalidate, and reinstall the item.

Deleting a User-Defined Tag Type (continued)

In a multiple-station system, you must assign the sysadmin role to the OSx station where you want to delete the new tag type. See the *SIMATIC PCS 7 OSx System Administration Manual* for details.

To remove a user-defined tag type from the OSx system, follow these steps:

1. Select **Controls->Change System State** from the main menu bar. Select **Offline** to set OSx to the Offline state.
2. Select **Tags->Remove from OSx->User-Defined Tag Type** from the main menu bar (Figure 4-23).
3. Click **Browse** to access a selection list of the user-defined tag types that have no configured instances. Select the tag type that you want to remove.
4. Click the **OK** buttons on the Select List and the Tag Installation dialog box to remove the tag type.



Operation Remove User-Defined Tag Type from OSx

Figure 4-23 Removing Tags

4.13 Creating Tag Details and Faceplates for New Tag Types

Restrictions on creating tag details and faceplates for user-defined tag types are listed below:

- You cannot use a Trend DDO.
- To display the status attribute in meaningful text rather than hexadecimal format, you must create animation files for it. See the Editors chapter of the *SIMATIC PCS 7 OSx Graphical Editor Manual*.

To create a tag detail for a new tag type, follow the steps below:

1. Select **Editors->Graphical Editor** from the OSx main menu bar. The graphical editor opens.
2. Select **Siemens File->Copy->Tag Detail** from the graphical editor menu bar. A list of tag details appears.
3. Double-click on the tag detail that is most like the one you want to create. The following prompt appears: **Copy Tag Detail <name> to**, where **<name>** is the tag detail that you selected.
4. Enter the name that you gave to the new tag type (for example, TANK). A format for the tag detail appears in the graphical editor window.
5. Add and delete DDOs to customize the tag detail. Tie the DDOs to your attributes with the Read Key editor.
6. Save and validate the tag detail.

Creating Tag Details and Faceplates for New Tag Types (continued)

To create a tag faceplate for the new tag type that can be displayed in a tag group, follow the steps below:

1. Select **Editors->Graphical Editor** from the OSx main menu bar. The graphical editor opens.
2. Select **Siemens File->Copy->Tag Group** from the graphical editor menu bar. A list of tag group faceplates appears.
3. Double-click on the tag group faceplate that is most like the one you want to create. The following prompt appears: **Copy Tag Group <name> to**, where **<name>** is the tag group faceplate that you selected.
4. Enter the name that you gave to the new tag type (for example, TANK). A format for the tag faceplate appears in the graphical editor window.
5. Add and delete DDOs to customize the tag faceplate. Tie the DDOs to your attributes with the Read Key editor.
6. Save and validate the tag group faceplate.

4.14 Testing the New Tag Type

Once the new tag type is installed, use the guidelines below to confirm that the tag type functions properly:

- If you configured the new tag type with alarm capability, create an alarm using the new tag type.
- Create a trend with an attribute that can be trended. See page 4-12.

With OSx in the Operate state, use STEP 7 CFC Debug to monitor and change the block parameters. Confirm that the values are displayed properly in the items listed below:

- Tag detail
- Tag group
- Alarm summary
- Real-time trend
- Tag Configurator (using the **Reload** button)

4.15 Error Messages

When you transfer a new tag type from the Engineering Toolset to OSx, a conversion log is generated. Table D-1 in Appendix D lists the error messages that can appear in this log.

Table D-4 in Appendix D shows error messages that may appear when you are installing, deleting, or using user-defined tag types.

Appendix A
Tag Configuration Status Report

Figure A-1 shows an example of the Configuration Status Report that is created whenever you install tags. You can view this report by entering y at the prompt when tag installation is complete.

```
                                                     13-May-98    Page 1

                         Configuration Status Report
                         Modified: 13-May-98   09:01am


NODE NAME      TAG NAME       TAG TYPE       TAG DESCRIPTION              OPERATION
------------   -----------    ----------     -------------------          -----------
C10            C10UNIT_3      UNIT           Dry Additive System          Installed
C10            C10DI_3        DI             vlv2_1 Close Limit Switch    Installed
C10            C10DO_5        DO             vlv21_5 Command              Installed
C10            C10VLV1_1      VLV1           Strawberry Flavor Feed Valve Installed
C10            C10CALC_0      CALC           Normal Cream Tank Level      Installed
C10            C10CALC_5      CALC           Walnuts Tank Level           Installed
C10            C10MTR1_4      MTR1           Cooling Vat 1 Discharge Pump Installed
C10            C10AO_5        AO             Flavor System Flush Pump Speed Installed
C10            C10LP_2        LOOP           Loop 2                       Installed
C10            C10TEXT_1      TEXT           Text Tag 1                   Installed
C10            C10TEXT_0      TEXT           Text Tag 0                   Installed
C10            C10AI_1        AI             Analog Alarm 1               Installed
C10            C10AI_2        AI             Analog Alarm 2               Installed
C10            C10DO10_0      DO10           DO10 Tag 0                   Installed
C10            C10CALC_1      CALC           Sweetener Tank Level         Installed
C10            C10IVAR_1      IVAR           Mixer 1 Temperature          Installed
C10            C10IVAR_0      IVAR           Mixer 1 Level                Installed
C10            C10IVAR_5      IVAR           Mixer 3 Temperature          Installed
C10            C10IVAR_4      IVAR           Mixer 3 Level                Installed
C10            C10IVAR_3      IVAR           Mixer 2 Temperature          Installed
C10            C10IVAR_2      IVAR           Mixer 2 Level                Installed
C10            C10DO10_5      DO10           DO10 Tag 5                   Installed
C10            C10DO10_4      DO10           DO10 Tag 4                   Installed
C10            C10DO10_3      DO10           DO10 Tag 3                   Installed
```

Figure A-1 Configuration Status Report

Figure A-2 shows an example of the S7 Tag Message Configuration section of the Configuration Status Report. The Operation column shows what action has been performed on the tag:

- Installed — The tag has been installed in the OSx database.
- Replaced — The tag data has been replaced by the new data in the installation file.
- Deleted — The tag has been deleted from the OSx database.
- No Change — The tag has not changed.
- None — The tag is in a different installation file and is not affected by this operation.

```
                                                            13–May–98    Page 6

                                   Configuration Status Report
                                   S7 Tag Message Configuration


TAG NAME       EV_ID       BLOCK NAME       DB NUMBER       TYPE          OPERATION
------------   ------      --------------   -------------   -----         -----------
C10unit_5      40375       UNIT             592             ALARM_S       No Change
C10mtr1_3      40354       MSN              568             ALARM_S       No Change
C10mtr1_0      40351       MSN              565             ALARM_S       Installed
C10area_4      40350       AREA             465             ALARM_S       Installed
C10area_0      40346       AREA             461             ALARM_S       Installed
C10do10_5      40283       DO10             386             ALARM_S       Replaced
C10ivar_5      40253       IVAR             356             ALARM_S       Installed
C10ivar_0      40248       IVAR             351             ALARM_S       Installed
C10ao_0        40224       AO               327             ALARM_S       Deleted
C10ao_3        40214       AO               317             ALARM_S       Installed
C10calc_5      40199       CALC             301             ALARM_S       Installed
C10calc_4      40198       CALC             300             ALARM_S       Installed
C3mtr1_5       40076       MSN              472             ALARM_S       None
C3vlv1_2       40020       VSN              484             ALARM_S       None
C3vlv1_1       40019       VSN              478             ALARM_S       None
```

Figure A-2 S7 Tag Message Configuration

Appendix B
Simulation Code Block FB200

```
FUNCTION_BLOCK FB200;
(* Simulation of a generic reactor or mixer vessel *)

(* Header *)

TITLE='VESSEL';
VERSION  :'1.0';
//KNOW_HOW_PROTECT-if uncommented
AUTHOR   :Siemens;
FAMILY   :SIMUL;
NAME     :Staff;

(* Definition of IN/OUT Variables *)

VAR_IN_OUT

V1_OP_CL:     BOOL;      //Valve 1 command
V2_OP_CL:     BOOL;      //Valve 2 command
M1_ON_OF:     BOOL;      //Motor 1 command
M2_ON_OF:     BOOL;      //Motor 2 command
RS_COOL:      BOOL;      //Reset temp to environment
V_RESET:      BOOL;      //Valve and motor reset

END_VAR;

(* Definition of INPUT only variables  *)

VAR_INPUT

V1_FLOW:     REAL := 25.0;      //Flowrate V1 0..100%
V2_FLOW:     REAL := 25.0;      //Flowrate V2
KFV:         REAL := 0.005;     //Valve Correction factor
V_HOT_FL:    REAL := 0.0;       //Flowrate Heat 0..100%
TMP_ENV:     REAL := 20.0;      //Environment Temperature
TMP_HOT:     REAL := 120.0;     //Product Temperature
T_LAG_IN:    REAL := 1000.0;    //Temp lag product
SAMPLE_T:    REAL := 1.0;       //Sample time in seconds

END_VAR;
```

```
(* Definition of OUTPUT only variables  *)

VAR_OUTPUT

OVERFLOW:       BOOL := FALSE;    //Overflow condition
LEVEL:          REAL := 0.0;      //Tank level in %
TMP_INT:        REAL := 20.0;     //Vessel Internal Temperature
FB_V1_OP:       BOOL;             //Valve1 Open Feedback
FB_V1_CL:       BOOL;             //Valve1 Closed Feedback
FB_V2_OP:       BOOL;             //Valve2 Open Feedback
FB_V2_CL:       BOOL;             //Valve2 Closed Feedback
FB_M1_ON:       BOOL;             //Motor 1 Run feedback
FB_M2_ON:       BOOL;             //Motor 2 Run feedback

END_VAR;

(*  Definition of internal use variables *)

VAR

NIV:                REAL := 0.0;        //Internal variable
TMP_IN1:            REAL := 20.0;       //Internal variable
TMP_IN2:            REAL := 20.0;       //Internal variable
T_LAG_IN_HILF:      REAL := 0.0;        //Internal variable
EXP_IN:             REAL := 1.0;
TMP_SH_IN:          REAL := 20.0;       //Shell temp internal
T_LAG_SH:           REAL := 10.0;
TMP_SH_HILF:        REAL := 20.0;
EXP_SHEL:           REAL := 1.0;
TMP_SHEL:           REAL := 20.0;       //Calculated shell temp
POS_V1:             REAL;               //Position V1 (0...100%)
POS_V2:             REAL;               //Position V2 (0...100%)
POS_M1:             REAL;               //Position M1 (0...100%)
POS_M2:             REAL;               //Position M2 (0...100%)

END_VAR;
```

```
BEGIN

// Cold Restart;

IF RS_COOL THEN;
    RS_COOL := FALSE;
    LEVEL := 0.0;
    TMP_SHEL := TMP_ENV;
    TMP_INT := TMP_ENV;
    TMP_IN1 := TMP_ENV;
    TMP_IN2 := TMP_ENV;
END_IF;

// Valve and Motor Reset;

IF V_RESET THEN;
    V_RESET := FALSE;
    V1_OP_CL := FALSE;
    V2_OP_CL := FALSE;
    M1_ON_OF := FALSE;
    M2_ON_OF := FALSE;
    POS_V1 := 0;
    POS_V2 := 0;
    POS_M1 := 0;
    POS_M2 := 0;
END_IF;

// Valve and Motor position and operation;

IF V1_OP_CL THEN;
    POS_V1 := POS_V1 + (25 * SAMPLE_T);
ELSE;
    POS_V1 := POS_V1 – (25 * SAMPLE_T);
END_IF;

IF V2_OP_CL THEN;
    POS_V2 := POS_V2 + (25 * SAMPLE_T);
ELSE;
    POS_V2 := POS_V2 – (25 * SAMPLE_T);
END_IF;
```

```
IF M1_ON_OF THEN;
    POS_M1 := POS_M1 + (25 * SAMPLE_T);
ELSE;
    POS_M1 := POS_M1 – (50 * SAMPLE_T);
END_IF;

IF M2_ON_OF THEN;
    POS_M2 := POS_M2 + (25 * SAMPLE_T);
ELSE;
    POS_M2 := POS_M2 – (50 * SAMPLE_T);
END_IF;

IF POS_V1 >= 100 THEN;
    POS_V1 := 100;
    FB_V1_OP := 1;
ELSE;
    FB_V1_OP := 0;
END_IF;

IF POS_V1 <= 0 THEN;
    POS_V1 := 0;
    FB_V1_CL := 1;
ELSE;
    FB_V1_CL := 0;
END_IF;

IF POS_V2 >= 100 THEN;
    POS_V2 := 100;
    FB_V2_OP := 1;
ELSE;
    FB_V2_OP := 0;
END_IF;

IF POS_V2 <= 0 THEN;
    POS_V2 := 0;
    FB_V2_CL := 1;
ELSE;
    FB_V2_CL := 0;
END_IF;
```

```
IF POS_M1 >= 100 THEN;
    POS_M1 := 100;
    FB_M1_ON := 1;
END_IF;

IF POS_M1 <= 0 THEN;
    POS_M1 := 0;
    FB_M1_ON := 0;
END_IF;

IF POS_M2 >= 100 THEN;
    POS_M2 := 100;
    FB_M2_ON := 1;
END_IF;

IF POS_M2 <= 0 THEN;
    POS_M2 := 0;
    FB_M2_ON := 0;
END_IF;

// Level simulation;

// This code performs the level simulation based on two valves, a fill valve
// and a drain valve. If the fill valve is open, the level of the vessel increases.
// If the drain valve is open, the level of the vessel decreases.

NIV := LEVEL;

NIV := NIV + (V2_FLOW * KFV * SAMPLE_T * POS_V2 / 100);
NIV := NIV – (V1_FLOW * KFV * SAMPLE_T * POS_V1 / 100);

IF NIV > 100.0 THEN;
    OVERFLOW := TRUE;
    NIV := 100.0;
ELSE;
    OVERFLOW := FALSE;
    IF NIV < 0.0 THEN;
        NIV := 0.0;
    END_IF;
END_IF;

LEVEL := NIV;
```

```
// Temperature Simulation

// Shell temperature simulation

TMP_SH_IN := (TMP_HOT - TMP_ENV) * V_HOT_FL /100.0 + TMP_ENV;
EXP_SHEL := 1.0 / EXP(1.0 * SAMPLE_T / T_LAG_SH);
TMP_SHEL := TMP_SH_IN + (TMP_SHEL-TMP_SH_IN) + EXP_SHEL;

// Product temperature simulation

IF V_HOT_FL > 0.0 THEN;
    T_LAG_IN_HILF := (T_LAG_IN / SAMPLE_T) * (LEVEL + 10.0) / 110.0;
    EXP_IN := 1.0 / EXP(1.0 / T_LAG_IN_HILF);
    TMP_IN1 := TMP_SHEL + (TMP_IN1-TMP_SHEL)*EXP_IN;
    TMP_IN2 := TMP_IN1 + (TMP_IN2 - TMP_IN1) * EXP_IN;
    TMP_INT := TMP_IN2;
END_IF;

END_FUNCTION_BLOCK;
```

Appendix C
User-defined Tank Tag

Retrieving the User Tags Library

This appendix lists the SCL source code for the TANK function block that is used as an example in Chapter 4. This function block is created to install a TANK tag on the OSx station. A library containing this function block is included on the CD-ROM labeled *SIMATIC PCS 7 OSx Library*. To install this library in the SIMATIC Manager, follow these steps:

1. Insert the CD-ROM labeled *SIMATIC PCS 7 OSx Library* in the CD-ROM drive.
2. From the SIMATIC Manager, select **File–>Retrieve**. The Retrieve – Select Archive dialog box appears.
3. In the Look In field, click the CD-ROM drive to select it.
4. Double-click the **APT** folder to list the archived files.
5. Select the **UserTags.arj** file and click **Open**. The Select Target dialog box appears.
6. Place the UserTags library in the S7 library or the S7 project of your choice.

You can now use the TANK function block in any S7 project that you create.

Placing enum_type Files on OSx

In order to install the TANK tag type in OSx, you must extract the **tkqual.ddl**, **tksts.ddl**, and **tkmode.ddl** files from the CD-ROM and place them in the /**usr/tistar/data/db/tables/local/enum_types** directory on OSx. Follow the steps below:

1. Insert the CD-ROM labeled *SIMATIC PCS 7 OSx Library* in the CD-ROM drive on your Windows NT computer.
2. Using Windows NT Explorer, copy the **.ddl** files from the APT directory on the CD-ROM to a diskette in the floppy disk drive.
3. Place the diskette in the disk drive on your OSx station.
4. Open an XTerm window and log in as `tistar`.
5. Enter `cd /usr/tistar/data/db/table/local/enum_types` to change to the **enum_types** directory.
6. Enter `mcopy -a a:tkqual.ddl tkqual.ddl` to copy the file from the disk to the hard drive. Repeat this step for **tksts.ddl** and **tkmode.ddl**.

These files are now available to define the parameters that use the keyword `enum_type`.

SCL Source Code for the TANK Function Block

(*---
Description
Source code for the Tank block (TANK)

Abstract
This file contains the source code for implementation of the TANK object on the S7 controllers. SCL version 4.0 or higher is required to correctly compile this object. The default values for this block assume that it will be attached to OB35.

This object is designed to show the techniques used in creating an OSx User Defined Tag Type.

Programming notes:

0. Before this block can be compiled by SCL, any OSx library blocks used within this block must be in the blocks folder (i.e. PACKSTAT and RBE_S).

1. The SYMBOL Table entry for TANK uses the COMMENT field to specify OSx block level parameters (e.g. ROWS=10).

2. Block parameters assigned with S7_m_c will be mapped into OSx.

3. Any block parameters which are assigned values within the block itself or can be changed from Osx are made unlinkable in CFC with S7_link=false.

4. Parameters which can be changed from OSx and also from within CFC is accomplished with three values: the OSx value, a PREVious OSx value, and a block value for CFC. To track which change is applicable, the following logic is applied.

```
If OSX value <> PREV Osx value
    Local value = OSx value
    PREV Osx value = OSx value
END
OSx value = local value
```

5. The ENUM_TYPE feature is demonstrated in this block. Bit patterns used for STATUS and other parameters' ALARM and LIMIT settings must agree with the corresponding specifications provided in the ”ddl” file. The ”ddl” files are available separately as individual files and must be placed in the OSx directory named /usr/tistar/data/db/local/enum_types. The ”ddl” files for tkmode.ddl, tkqual.ddl, and tksts.ddl are embedded within this SCL file for documentation purposes.

6. Once this block has been compiled successfully in SCL, next you must complete the RBE_S message configuration by selecting the TANK block in the blocks folder and then specifying its settings under SPECIAL OBJECT PROPERTIES->MESSAGE.

This object maps to OSx as an TANK type with the following tag attributes:
STATUS
GAUGE_LEV
HH_ALM
H_ALM
L_ALM
LL_ALM
H_RANGE
L_RANGE
TEMP
HH_TEMP
H_TEMP
L_TEMP
LL_TEMP

```
-------------------------------------------------------*)

FUNCTION_BLOCK ”TANK” {S7_m_c := ’true’}
AUTHOR : Customer;
NAME   : TANK;
FAMILY : CONTROL;
VERSION: ’4.0’;
// KNOW_HOW_PROTECT;

VAR_INPUT
 // TIP : Put any OSx ”NONETWORK” parameters first and
//  specify an initial value to be used in OSx.
```

```
                    CHANGE   { S7_m_c     := ’true’;
                             S7_link    := ’false’;
                             S7_visible := ’false’} :REAL := 1.0 ;//@NONETWORK

                         EV_ID { S7_m_c     := ’true’;
                             S7_link    := ’false’;
                             S7_visible := ’false’;
                             S7_param   := ’false’;
                             S7_server  := ’alarm_archiv’;
                             S7_a_type  := ’alarm_s’ }
                             :DWORD     := 0;//Identity number for RBE_S
                                                   message

END_VAR

VAR_IN_OUT
    // TIP: SCL does not permit default initialization of IN_OUT parameters.
    //      However, once the block has been compiled, the block I/O
    //      can be given default values in the blocks folder by selecting the
    //      LAD/STL editor and then opening the block and then
    //      editing the ”Initial Value”.
    //

    // --- OSx interface variables : Required for an RBE_S message.
    // --- The AND_MASK, OR_MASK,  STATUS, and HEALTH must be contiguous.
    //
    AND_MASK { S7_m_c     := ’true’;
               S7_link    := ’false’;
               S7_visible := ’false’}
              :WORD;                // OSx AND mask

    OR_MASK  { S7_m_c     := ’true’;
               S7_link    := ’false’;
               S7_visible := ’false’}
              :WORD;                // OSx OR mask

    STATUS   { S7_m_c     := ’true’;
               S7_link    := ’false’;
               S7_visible := ’false’}
              :WORD;                // OSx STATUS @ENUM_TYPE=tkqual READONLY
```

```
// STATUS is a ”bitmask” constructed by this block. In OSx
// the tkqual.ddl file is defined:
//
// DEFINE tkqual AS STATUS OF
//      (HH_ALM = 0x0800,
//       H_ALM  = 0x0400,
//       L_ALM  = 0x0200,
//       LL_ALM = 0x0100,
//       L_TEMP = 0x0080,
//       H_TEMP = 0x0040,
//       HH_TEMP= 0x0020,
//       LL_TEMP= 0x0010);
//

HEALTH   { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :WORD;                // OSx Health (ALARM_S ”Error” output)

TANK_MD  { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :INT;                 // OSx TANK_MD @ENUM_TYPE=tkmode

// TANK_MD is a value input for this block. In OSx
// the tkmode.ddl file is defined:
//
// DEFINE tkmode AS (
// OOS      =999,
// MANUAL   = 10,
// RUN      =  9,
// FILL     =  8,
// EMPTY    =  7,
// CLOSE    =  6,
// RAISE    =  5,
// HALT     =  4,
// LOW      =  3,
// SEARCHH2O=  2,
// ABORTH2O =  1,
// NOMODE   =  0);
//

TANK_STS { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :WORD;                // @ENUM_TYPE=tksts READONLY
```

```
// TANK_STS is a ”bitmask” constructed by this block. In OSx
// the tksts.ddl file is defined:
//
// DEFINE tksts AS SET OF
//(OOS      = 0x8000,
// MANUAL   = 0x4000,
// RUN      = 0x2000,
// FILL     = 0x1000,
// EMPTY    = 0x0800,
// CLOSE    = 0x0400,
// RAISE    = 0x0200,
// HALT     = 0x0100,
// LOW      = 0x0080,
// SEARCHH2O= 0x0040,
// ABORTH2O = 0x0020);

TEMP     { S7_m_c     := ’true’;
           S7_visible := ’true’}
          :REAL;                // @ALARM=0x00f0 READONLY

GAUGE_LEV{ S7_m_c     := ’true’;
           S7_visible := ’true’}
          :REAL;                // @ALARM=0x0f00 DEADBAND READONLY

H_RANGE  { S7_m_c     := ’true’;
           S7_visible := ’true’}
          :REAL;                // @UPLOAD READONLY

L_RANGE  { S7_m_c     := ’true’;
           S7_visible := ’true’}
          :REAL;                // @UPLOAD READONLY

// The values for HH_ALM, H_ALM, L_ALM, LL_ALM are used to set a
// bit in STATUS (indicated with LIMIT) when GAUGE_LEV is in the
// indicated alarm condition. The GAUGE_LEV specifier ”ALARM” is used
// to define its use in an OSx alarm message for the specified alarm
// bits (in STATUS). Thus, ALARM=0x0f00 means it applies for the
// alarms of 0x0800 (HH_ALM), 0x0400 (H_ALM), 0x0200 (L_ALM) and
// 0x0100 (LL_ALM).
//

HH_ALM   { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :REAL;                // @LIMIT=0x0800,CRIT
```

```
H_ALM    { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :REAL;                // @LIMIT=0x0400,WARN

L_ALM    { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :REAL;                // @LIMIT=0x0200,WARN

LL_ALM   { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :REAL;                // @LIMIT=0x0100,CRIT

// The values for L_TEMP, H_TEMP, HH_TEMP, LL_TEMP are used to set a
// bit in STATUS (indicated with LIMIT) when TEMP is in the
// indicated alarm condition. The TEMP specifier ”ALARM” is used
// to define its use in an OSx alarm message for the specified alarm
// bits (in STATUS). Thus, ALARM=0x00f0 means it applies for the
// alarms of 0x0080 (L_TEMP), 0x0040 (H_TEMP), 0x0020 (HH_TEMP) and
// 0x0010 (LL_TEMP).
//
L_TEMP   { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :REAL;                // @LIMIT=0x0080,WARN

H_TEMP   { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :REAL;                // @LIMIT=0x0040,WARN

HH_TEMP  { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :REAL;                // @LIMIT=0x0020,CRIT

LL_TEMP  { S7_m_c     := ’true’;
           S7_link    := ’false’;
           S7_visible := ’false’}
          :REAL;                // @LIMIT=0x0010,CRIT
```

```
   // --- CFC variables ---
   HHA      :REAL    ; // High-high gauge level limit
   HA       :REAL    ; // High gauge level limit
   LA       :REAL    ; // Low gauge level limit
   LLA      :REAL    ; // Low-low gauge level limit
   LADB     :REAL    ; // Level Alarm deadband

   HHA_TEMP :REAL    ; // High-high temp limit
   HA_TEMP  :REAL    ; // High temp limit
   LA_TEMP  :REAL    ; // Low temp limit
   LLA_TEMP :REAL    ; // Low-low temp limit
   TADB     :REAL    ; // Temp Alarm deadband

   MODE      :INT;     // CFC (not OSx) MODE of TANK.

END_VAR

VAR_OUTPUT

   // --- State variables ---
   INHHA    :BOOL :=0;   //  Tank level in high-high (GAUGE_LEV > HHA)
   INHA     :BOOL :=0;   //  Tank level in high alarm(GAUGE_LEV > HA)
   INLA     :BOOL :=0;   //  Tank level in low alarm (GAUGE_LEV < LA)
   INLLA    :BOOL :=0;   //  Tank level in low-low alarm (GAUGE_LEV < LLA)
   INHHTEMP :BOOL :=0;   //  Tank temp in high-high alarm(TEMP > HHA_TEMP)
   INHTEMP  :BOOL :=0;   //  Tank temp in high alarm     (TEMP > HA_TEMP)
   INLTEMP  :BOOL :=0;   //  Tank temp in low alarm      (TEMP < LA_TEMP)
   INLLTEMP :BOOL :=0;   //  Tank temp in low-low alarm  (TEMP < LLA_TEMP)
   OOS      :BOOL :=0;   //  Tank in Out of Service
   MANUAL   :BOOL :=0;   //  Tank in Manual Mode
   RUN      :BOOL :=0;   //  Tank in Run Mode
   FILL     :BOOL :=0;   //  Tank in Fill Mode
   EMPTY    :BOOL :=0;   //  Tank in Empty Mode
   CLOSE    :BOOL :=0;   //  Tank in Closed Mode
   RAISE    :BOOL :=0;   //  Tank in Raise Mode
   HALT     :BOOL :=0;   //  Tank in Halt
   LOWER    :BOOL :=0;   //  Tank in Lower float Mode
   SEARCHH2O:BOOL :=0;   //  Tank in Search Water Mode
   ABORTH2O :BOOL :=0;   //  Tank in Abort Search
END_VAR


CONST
// --- REQUIRED RBE ATTACHMENT TYPES (for RBE_S) ---
   C_RBE_PKBOOL      := 1;  // Packed boolean attachment
   C_RBE_PKBOWO      := 2;  // Packed bool and word attachment
END_CONST
```

```
VAR
// --- Required OSx Prior call values for RBE_S usage ---
   PRIOR_STATUS  :WORD;             // Value of STATUS on last call
   PRIOR_AND_MASK:WORD;             // Value of AND_MASK on last call
   PRIOR_OR_MASK :WORD;             // Value of OR_MASK on last call
   RBE_S_1       :RBE_S;            // RBE_S instance
   RBE_EXCPT     :BOOL;             // RBE exception flag

// --- Required OSx Prior call values for OSx command values ---
   PREV_TANK_MD  :INT;              // Value of TANK_MD on last call

   PREV_HH_ALM   :REAL;             // Value of HH_ALM on last call
   PREV_H_ALM    :REAL;             // Value of H_ALM on last call
   PREV_L_ALM    :REAL;             // Value of L_ALM on last call
   PREV_LL_ALM   :REAL;             // Value of LL_ALM on last call

   PREV_HHTMP    :REAL;             // Value of HH_TEMP on last call
   PREV_HTEMP    :REAL;             // Value of H_TEMP on last call
   PREV_LTEMP    :REAL;             // Value of L_TEMP on last call
   PREV_LLTEMP   :REAL;             // Value of LL_TEMP on last call


END_VAR

BEGIN
   // --- Reconcile OSx command values to CFC values ---
   //
   // Here a check is made to see if a new value has been commanded
   // from OSx.
   //
   IF (TANK_MD <> PREV_TANK_MD) THEN  // Check TANK_MD from OSx
       MODE        := TANK_MD;
       PREV_TANK_MD:= TANK_MD;
   END_IF;
   TANK_MD := MODE;

   IF (HH_ALM <> PREV_HH_ALM) THEN    // New HHA on delta
       HHA         := HH_ALM;
       PREV_HH_ALM := HH_ALM;
   END_IF;
   HH_ALM := HHA;

   IF (H_ALM <> PREV_H_ALM) THEN      // New HA on delta
       HA         := H_ALM;
       PREV_H_ALM := H_ALM;
   END_IF;
```

```
H_ALM := HA;

IF (L_ALM <> PREV_L_ALM) THEN      // New LA on delta
    LA         := L_ALM;
    PREV_L_ALM := L_ALM;
END_IF;
L_ALM := LA;

IF (LL_ALM <> PREV_LL_ALM) THEN    // New LLA on delta
    LLA         := LL_ALM;
    PREV_LL_ALM := LL_ALM;
END_IF;
LL_ALM := LLA;

IF (HH_TEMP <> PREV_HHTMP) THEN
    HHA_TEMP   := HH_TEMP;
    PREV_HHTMP := HH_TEMP;
END_IF;
HH_TEMP := HHA_TEMP;

IF (H_TEMP <> PREV_HTEMP) THEN
    HA_TEMP    := H_TEMP;
    PREV_HTEMP := H_TEMP;
END_IF;
H_TEMP := HA_TEMP;

IF (L_TEMP <> PREV_LTEMP) THEN
    LA_TEMP    := L_TEMP;
    PREV_LTEMP := L_TEMP;
END_IF;
L_TEMP := LA_TEMP;

IF (LL_TEMP <> PREV_LLTEMP) THEN
    LLA_TEMP    := LL_TEMP;
    PREV_LLTEMP := LL_TEMP;
END_IF;
LL_TEMP := LLA_TEMP;


// Now check for LEVEL alarms: high-high, high, low, and low-low alarms.
// Note: LADB is used as an ”Alarm DeadBand” to prevent alarm chatter.
//
IF (GAUGE_LEV > HHA) THEN            // in high-high alarm
    INHHA := TRUE;
END_IF;

IF (GAUGE_LEV <= (HHA - LADB)) THEN  // out of high-high alarm
    INHHA := FALSE;
END_IF;
```

```
IF (GAUGE_LEV < LLA) THEN            // in low-low alarm
    INLLA := TRUE;
END_IF;

IF (GAUGE_LEV >= (LLA + LADB)) THEN  // out of low-low alarm
    INLLA := FALSE;
   END_IF;

IF (GAUGE_LEV > HA) THEN             // in high alarm
    INHA := TRUE;
END_IF;

IF (GAUGE_LEV <= (HA - LADB)) THEN   // out of high alarm
    INHA := FALSE;
END_IF;

IF (GAUGE_LEV < LA) THEN             // in low alarm
    INLA := TRUE;
END_IF;

IF (GAUGE_LEV >= (LA + LADB)) THEN   // out of low alarm
    INLA := FALSE;
END_IF;

// Now check for TEMP alarms: high-high, high, low, and low-low alarms.
// Note: TADB is used as an ”Alarm DeadBand” to prevent alarm chatter.
//
IF (TEMP > HHA_TEMP) THEN            // in high-high alarm
    INHHTEMP := TRUE;
END_IF;

IF (TEMP <= (HHA_TEMP - TADB)) THEN  // out of high-high alarm
    INHHTEMP := FALSE;
END_IF;

IF (TEMP < LLA_TEMP) THEN            // in low-low alarm
    INLLTEMP := TRUE;
END_IF;

IF (TEMP >= (LLA_TEMP + TADB)) THEN  // out of low-low alarm
    INLLTEMP := FALSE;
END_IF;
```

```
IF (TEMP > HA_TEMP) THEN             // in high alarm
    INHHTEMP := TRUE;
END_IF;

IF (TEMP <= (HA_TEMP - TADB)) THEN   // out of high alarm
    INHTEMP := FALSE;
END_IF;

IF (TEMP < LA_TEMP) THEN             // in low alarm
    INLTEMP := TRUE;
END_IF;

IF (TEMP >= (LA_TEMP + TADB)) THEN   // out of low alarm
    INLTEMP := FALSE;
END_IF;

// -- Pack STATUS variable (bits must agree with tkqual.ddl )
STATUS   := PACKSTAT (B1:=0,B2:=0,B3:=0,B4:=0,
             B5 := INHHA,
             B6 := INHA,
             B7 := INLA,
             B8 := INLLA,
             B9 := INLTEMP,
             B10:= INHTEMP,
             B11:= INHHTEMP,
             B12:= INLLTEMP,
             B13:=0,B14:=0,B15:=0,B16:=0);

 // -- Pack TANK_STS variable (bits must agree with tksts.ddl)

OOS      := (MODE = 999);
MANUAL   := (MODE = 10);
RUN      := (MODE = 9);
FILL     := (MODE = 8);
EMPTY    := (MODE = 7);
CLOSE    := (MODE = 6);
RAISE    := (MODE = 5);
HALT     := (MODE = 4);
LOWER    := (MODE = 3);
SEARCHH2O:= (MODE = 2);
ABORTH2O := (MODE = 1);
```

```
   TANK_STS := PACKSTAT (
                B1 := OOS,
                B2 := MANUAL,
                B3 := RUN,
                B4 := FILL,
                B5 := EMPTY,
                B6 := CLOSE,
                B7 := RAISE,
                B8 := HALT,
                B9 := LOWER,
                B10:= SEARCHH2O,
                B11:= ABORTH2O,
                B12:=0,B13:=0,B14:=0,B15:=0,B16:=0);


    // --- RBE_S will generate and control the exception ---
    RBE_S_1(TYP      := C_RBE_PKBOOL,
            EV_ID    := EV_ID,
            STATUS   := STATUS,
            AND_MASK := AND_MASK,
            OR_MASK  := OR_MASK,
            HEALTH   := HEALTH,
            P_STAT   := PRIOR_STATUS,
            P_AND    := PRIOR_AND_MASK,
            P_OR     := PRIOR_OR_MASK,
            RBE_EXCPT:= RBE_EXCPT);

END_FUNCTION_BLOCK;
```

Appendix D Troubleshooting

D.1 Error Messages in the Conversion Log

When you transfer tags from the Engineering Toolset to OSx, a conversion log is generated. Table D-1 lists the error messages, Table D-2 lists the warning messages, and Table D-3 lists the information messages that can appear in this log.

If an error message appears in the conversion log report that is not mentioned in the following tables, check the Engineering Toolset software components to ensure that they are of the correct revision level as described in the installation instructions on pages 1-4 and following. If the Toolset software is not up-to-date, install the correct version of the software and repeat the process of mapping and transferring data to OSx, then retry the operation on OSx that resulted in the error. At this point, if the problem persists, contact the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800–333–7421. In other countries, call 49–911–895–7000.

Table D-1 Conversion Log Error Messages

Error Message	Explanation
Assigned value out-of-range: assignment FB: fb_name	The value given in the ROWS assignment shown in the message is out of range. The range of acceptable values is 1 to 500. The type name of the FB where the assignment appeared is given by fb_name. Example: **Assigned value out-of-range: ROWS=999 FB: FBZZ**
Bit name too long: assignment; fb_name:parameter_name	The bit name shown in the assignment is longer than the 10-character limit imposed by the OSx database. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Bit name too long: \extra_long_name\B13; FBXX:MODE**
Bit number out of range: bit_name_assignment	The bit number in the bit_name_assignment shown in the message is out of range. The maximum available bit number is 16. The FB type where the assignment was found will be listed as an unknown block type in a warning message on a preceding line in the conversion log. Example: **Bit number out of range: \idle\B17**
Could not append data to install.tag file	An error occurred while trying to write tag install data to disk. The most likely cause of this problem is lack of free disk space in the **/usr/tistar** file system. Regular archiving of trend data will usually avoid this problem.
Could not append title to install.tag file	An error occurred while trying to write tag install data to disk. The most likely cause of this problem is lack of free disk space in the **/usr/tistar** file system. Regular archiving of trend data will usually avoid this problem.
Data node not configured: plc_name	The controller named in the message is not configured in the OSx database. Before tag information generated by the Engineering Toolset can be processed on OSx, the controller designated by the program name in the Engineering Toolset data must be configured in OSx. Example: **Data node not configured: C101**
Data node not S7 type: plc_name	The controller named in the message is not an S7 controller. Check the control node configuration on OSx to ensure that the controller has been configured correctly. Example: **Data node not S7 type: C101**
Duplicate tag type name: tag_type_name; fb_name, fb_name	The two FBs named in the message both intend to use the same tag_type_name for a new tag type to be installed in OSx. This results from identical RENAME assignments in each of the two FB types. Change the tag_type_name in one of the RENAME assignments to resolve the problem. Example: **Duplicate tag type name: XTAG; FB001, FB002**

Table D-1 Conversion Log Error Messages (continued)

Error Message	Explanation
Enumerated type already defined: fb_name: parameter_name	An enumerated data type can be assigned to an attribute using either ENUM_TYPE assignments, bit name assignments, or enumeration member assignments. Only one of the three methods can be used for a given attribute. This message indicates that some combination of the methods was employed for the FB type and parameter shown in the message. Change the parameter assignments so that only one method is used. Example: **Enumerated type already defined: FBZZ:STATUS**
Enum type name too long: assignment; fb_name: parameter_name	The enumerated type name shown in the assignment exceeds the 12-character limit imposed by the OSx database. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Enum type name too long: ENUM_TYPE=extra_long_name; FBXX:STATUS**
Enum value out of range: enum_name_assignment	The value specified in the enum_name_assignment is out of range. The acceptable range of values is -32768 to 65535. The value may be expressed as either a decimal or hexadecimal number. The FB type where the assignment was found will be listed as an unknown block type in a warning message on a preceding line in the conversion log. Example: **Enum value out of range: \done\99999**
FB parameter does not map to attribute: fb_name: parameter_name	For an FB type that is to be mapped to an existing OSx tag type, this message appears for each parameter of the FB type that does not map to an attribute of the OSx database table that represents the tag type. Two options are available for resolving this problem: 1. Ensure that the name of each mapped parameter corresponds to the name of an attribute in the OSx database table for the tag type. 2. Use the NOTOSX option to tell OSx to ignore the parameter so that it will not attempt to map it to an attribute in the database table. Example, where FBDI is to be mapped to the DI tag type: **FB parameter does not map to attribute: FBDI:STATE**
Invalid assigned value: FB: fb_name	One or more of the PLC type names in the assignment shown were not recognized by the system. The valid type names are S7-414, S7-416, and S7-417. The type name of the FB where the assignment appeared is given by fb_name. Example: **Invalid assigned value: PLC=S7_416 FB: FBXX**
Field overrun	A text field in the transferred mapper data exceeded the maximum size expected by OSx. This error usually follows some other error that causes OSx to be unable to continue processing. Resolving the cause of the first error should eliminate both problems. Example: **Field overrun: 12345~!#$%)(*&^%'/?}{ABCDEF@TGN=SCALED_INT60**
Invalid attribute name: assignment; fb_name: parameter_name	In the ATTR assignment shown in the message, the attribute name is longer than the 19-character limit imposed by the OSx database. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Invalid attribute name: ATTR=this_is_a_very_long_name; FBXX:VALUE**

Table D-1 Conversion Log Error Messages (continued)

Error Message	Explanation
Invalid bit assignment format: discrete_assignment	The discrete_assignment (bit name or enumeration member name assignment) shown in the message was found to be missing the second backslash character. This character is required in order to determine the correct value for the bit or enumeration member. Examples: **Invalid bit assignment format: \running** **Invalid bit assignment format: \offB01**
Invalid bit number symbol: assignment; fb_name: parameter_name	The LIMIT assignment shown in the message will not yield a valid bit number. Correct the assigned value so that it adheres to the format for alarm limit assignments as described in Section 4.6 of this manual. Example: **Invalid bit number symbol: LIMIT=K000; MY_BLOCK:STATE**
Invalid bit number symbol: bit_name_assignment	The bit_name_assignment shown will not yield a valid bit number. The FB type where the assignment was found will be listed as an unknown block type in a warning message on a preceding line in the conversion log. Correct the assignment so that it adheres to the format for STATUS and other bit-oriented attributes described in Section 4.5 of this manual. Example: **Invalid bit number symbol: \off\BX**
Invalid data type for status: assignment; fb_name: parameter_name	The data type of the parameter named in the bit name assignment shown in the message is not appropriate for an OSx STATUS attribute. A parameter must be typed as a WORD in SCL in order to function properly in this role. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Invalid data type for status: \data_val\B1; FBZZ:STATUS**
Invalid enum value symbol: enum_name_assignment	The enumeration member assignment shown in the message will not yield a usable value. The value portion of the assignment must be a number expressed in decimal or hexadecimal notation. The FB type where the assignment was found will be listed as an unknown block type in a warning message on a preceding line in the conversion log. Example: **Invalid enum value symbol: \finished\#25**
Invalid tag name TGN=tag_name	The tag_name in the TGN assignment is longer than the 12-character limit imposed by the OSx database. Shorten the name and try again. Example: **Invalid tag name: TGN=my_extra_long_tag_name**
Invalid tag type name: tag_type_name; fb_name	The tag_type_name shown in the message is not usable as a name for a tag type in OSx for one of the following reasons: 1. The name is longer than the 4-character limit imposed by the OSx database. 2. The name is already in use as a tag type name in the OSx database. Modify the name as appropriate to resolve the problem. The FB named in the message is the type name of the FB where a RENAME assignment specifying the offending tag_type_name was encountered. Example: **Invalid tag type name: AI; MY_AI_BLOCK**

Table D-1 Conversion Log Error Messages (continued)

Error Message	Explanation
Member name too long: assignment; fb_name: parameter_name	The name for the enumeration member shown in the assignment is longer than the 12-character limit imposed by the OSx database. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Member name too long: \extra_long_name\44; FBXX:STATE**
Message block type detection failed for FB: fb_name	This message appears if one of the mapper files generated on the engineering station has been removed from OSx. Repeat the file transfer from the engineering station to OSx and try again. If the problem persists, call the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421. In other countries, call 49-911-895-7000. Example: **Message block type detection failed for FB: FB88**
Missing range attribute: fb_name	For the FB type named in the message, one or more parameters were declared to be bounded using the DEADBAND option. However, either one or both of the parameters H_RANGE and L_RANGE were found to be missing from the FB's parameter list. Either remove the DEADBAND option from the FB's parameters or add the H_RANGE and L_RANGE parameters. Example: **Missing range attribute: FBXX**
No alarm attribute for limit: fb_name:parameter_name LIMIT=0x0000	For the FB type named in the message, no parameter could be found having an ALARM assignment which includes the bit in the LIMIT bit pattern shown in the message. All bit patterns in LIMIT assignments must have a corresponding ALARM bit for some parameter of the FB. Either add the LIMIT bit to an ALARM assignment or remove the LIMIT assignment from the parameter named in the message. Example: **No alarm attribute for limit: FBYY:HH_ALM LIMIT=0x0400**
No value assigned: assignment; fb_name:parameter_name	The value part of an ENUM_TYPE, ALARM, or LIMIT assignment is missing. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **No value assigned: ALARM=; FBXX:STATUS**
Reserved status bit: assignment; fb_name:parameter_name	The bit number or alarm bit pattern shown in the assignment references at least one of the bit positions reserved for special use in OSx STATUS attributes. This problem can occur in a bit name assignment for a STATUS attribute or in the bit pattern of an ALARM assignment. The reserved STATUS bits (B14, B15, or B16 for bit names; 0x0007 for ALARM bit patterns) must not be used in assignments affecting the STATUS attribute. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Examples: **Reserved status bit: \stopped\B14; FBZZ:STATUS** **Reserved status bit: ALARM=0x0004; FBYY:CONDITION**

Table D-1   Conversion Log Error Messages (continued)

Error Message	Explanation
Tag data write error tag_name	An error occurred while trying to write install data to disk for the named tag. The most likely cause of this problem is lack of free disk space in the **/usr/tistar** file system. Regular archiving of trend data will usually avoid this problem.
Tag type assignment overridden: tag_type_name; fb_name	This message appears if more than one RENAME or TAG_TYPE assignment is detected in the comment field for the FB type named in the message. The tag_type_name in the message will be discarded and the name from the last assignment processed will be retained. To resolve the problem, make only one RENAME or TAG_TYPE assignment for each FB type. See Section 4.3 of this manual for details. Example: **Tag type assignment overridden:XYZ1; FBZZ105**
Unacceptable version string: version_number	The Engineering Toolset mapper version_number displayed in the message is incompatible with this release of OSx. Make sure the Engineering Toolset software components are up-to-date as described in the installation instructions (pages 1-4 and following).
Undefined enum type: assignment; fb_name: parameter_name	A definition for the enumerated data type named in the assignment could not be found. In order to use the data type, the definition must exist either in the OSx database or in a source file provided by the user. See Section 4.5 of this manual for details. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Undefined enum type: ENUM_TYPE=my_own_qual; FBYY:STATUS**
Unrecognizable value string: assignment FB: fb_name	The block assignment shown in the message will not yield a usable value. The value portion of the assignment must be a number expressed in decimal or hexadecimal notation. The type name of the FB where the assignment appeared is given by fb_name. Example: **Unrecognizable value string: ROWS=#400 FB: FBYY**
Undelimited string	Undelimited string: This usually results from an unescaped double-quote mark in a Comment field. In order for OSx to correctly process a double-quote mark that is to be included in a tag description or assignment, the mark must be "escaped" by entering two marks in immediate succession (that is, ””). Example: **Undelimited string: 12345~!@#$%)(*&^%‘’/**
Unmatched delimiter in comment	This usually results from an unmatched single-quote mark in the assignment portion of a comment field. The marks are used to delimit character strings in initial value assignments for tag attributes. The assignment portion of the Comment field is the part of the field that follows the first ”at” sign (@). To avoid this error, make sure to use single-quote marks in matched pairs when entering a character string assignment. Example: **Unmatched delimiter in comment: @TEXT_1=’Whatever STATUS=A**

Error Messages in the Conversion Log (continued)

Table D-1 Conversion Log Error Messages (continued)

Error Message	Explanation
Unrecognized criticality level: assignment; fb_name: parameter_name	The alarm criticality level in the LIMIT assignment shown in the message is not valid. The criticality part of the assignment must be INFO, WARN, or CRIT. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Unrecognized criticality level: LIMIT=0x1000,BAD; FBXX:STATE**
Unrecognized FB assignment: assignment FB: fb_name	The keyword in the block assignment shown in the message is invalid. See Section 4.3 of this manual for a description of the block assignments supported by the system. The type name of the FB where the assignment appeared is given by fb_name. Example: **Unrecognized FB assignment: ROWSS=111 FB: FBYY**
Unrecognized parameter assignment: assignment; fb_name:parameter_name	The assignment shown in the message is not a valid assignment for a block parameter. For a description of parameter assignments supported by the system, see Section 4.5 of this manual. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Unrecognized parameter assignment: ENUM=mine; FBZZ:STATUS**
Unrecoverable error	An unexpected error condition has occurred. Contact the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421. In other countries, call 49-911-895-7000.
Wrong assignment format: assignment; fb_name: parameter_name	The format of the assignment shown in the message is incorrect for a value assignment for the named parameter. The correct format for the assignment is: KEYWORD=value, where acceptable choices for KEYWORD and value are explained in Chapter 4 of this manual. Example: **Wrong assignment format: ENUM_TYPE; FBXX:STATUS**
Wrong data type for bits: assignment; fb_name: parameter_name	The data type of the parameter named in the message is not compatible with the bit-oriented data type associated with the assignment. A parameter must be typed as WORD, DWORD, INT, or DINT in SCL in order to function properly as a bit-oriented data type in the OSx database. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Wrong data type for bits: \hold\B12; PID:SETPOINT**
Wrong data type for enumeration: assignment; fb_name: parameter_name	The data type of the parameter named in the message is not compatible with the enumerated data type associated with the assignment. A parameter must be typed as WORD in SCL in order to function properly as an enumerated type in the OSx database. The names of the FB type and parameter where the assignment appeared are given by fb_name and parameter_name, respectively. Example: **Wrong data type for enumeration: \hot\999; PID:SETPOINT**

Table D-2 Conversion Log Warning Messages

Error Message	Explanation
Alarm limit ignored: fb_type_name:parameter_name LIMIT=0x0000	The total number of LIMIT flags for a single FB type is limited to 10. This message appears if more than 10 different flag bits are defined by the LIMIT assignments for the parameters of the FB type. The hexadecimal number in the message indicates which flag will be ignored. To resolve the problem, eliminate one or more flag bits from the LIMIT and ALARM assignments for the FB type. Example: **Alarm limit ignored: ALRM:STATUS LIMIT=0x0040**
Attribute name assignment overridden: attr_name	This message appears if multiple ATTR assignments are given for a single parameter. The attribute name in the message is overridden by the attribute name in the last ATTR assignment encountered. Example: **Attribute name assignment overridden: MY_PARAMETER**
Chart name repeated within PV name: pv_name	An element of the process variable (PV) name is a duplicate of the CFC chart name. This situation may arise if an FB instance is given a name that incorporates the name of the CFC chart container. This creates problems for determination of the tag name to be used for the FB instance. Change the FB instance name so that it does not include a copy of the CFC chart name and try again. Example: **Chart name repeated within PV name: C101_CFC1_CFC1_ao**
Description truncated for tag:: tag_name	The description for the named tag was truncated because it exceeded the 30-character limit imposed by the OSx database. Example: **Description truncated for tag: my_tag**
Duplicate tag name: tag_name	Two or more FB instances have been assigned the tag_name given in the message. In order to properly install as tags on OSx, all FB instance names must be unique. Example: **Duplicate tag name: XXX_101**
Found no match for this tag in message store: tag_name	Messages of this form appear when processing Engineering Toolset data that contains references to FB types that have yet to be installed as new tag types on OSx. In this case, the message should be ignored. If the tag names in the messages are associated with FB types that are already defined in the OSx database, then there may be a mismatch between the OSx software release level and that of the Engineering Toolset. Make sure the Engineering Toolset software components are up-to-date as described in the installation instructions (pages 1-4 and following). Example: **Found no match for this tag in message store: my_tag**
Invalid attribute name in assignment: pv_name: bad_attr_name	In the Comment field of an FB instance, an assignment of the form: `attribute_name=initial_value` was encountered where the attribute_name did not match the name of any of the attributes of the associated tag type table in the OSx database. The FB instance is identified by the process variable name (pv_name) from the Engineering Toolset; the offending attribute name is given by bad_attr_name. The pv_name includes all the elements of the naming hierarchy used in the Engineering Toolset. Example: **Invalid attribute name in assignment: C101_CFC1_ao_101:MOOD**

Table D-2 Conversion Log Warning Messages (continued)

Error Message	Explanation
Requested table size exceeds initial limit: fb_type_name	The value in the ROWS assignment for the named FB type exceeds the limit of 500. Reduce the value and try again. The capacity for a user–defined tag type can be enlarged beyond the 500-row limit using the Tag Capacity utility after the new tag type is installed. Example: **Requested table size exceeds initial limit: MYFB**
Truncated function block type name: original_name -> truncated_name	Names of FB types are limited to 12 characters in order to comply with OSx internal constraints. If OSx encounters an FB type name longer than this, it is truncated for internal use as the database table name for the user-defined tag type. Example: **Truncated function block type name: MYFBTYPENAMEXX -> MYFBTYPENAME**
Unknown function block type: fb_name	The named FB does not exist in the OSx database and will be treated as a candidate for a user-defined tag type. Example: **Unknown function block type: WXYZ**
Unknown tag type: tag_type_name, FB: fb_type_name	The tag_type_name specified in a TAG_TYPE assignment could not be found in the OSx database. The FB type where the assignment was found is given by fb_type_name. Correct the tag type name and try again. Example: **Unknown tag type: POOL, FB: MY_PID_FB**

Table D-3 Conversion Log Information Messages

Error Message	Explanation
Data type defined in file and DB, using DB definition: enum_type_name	This message appears if the enumerated data type in an ENUM_TYPE assignment is found to exist both in a source file and in the database. This happens if a data type is defined in a source file and is used to install more than one user-defined tag type. In such cases, the database definition for the enumerated data type is used. Example: **Data type defined in file and DB, using DB definition: my_qual**
New user-defined tag types available for installation:	This message serves as a heading line for the list of tag types available for installation which follows it.
No new S7 user-defined tag types present	The message appears when no user-defined FB types are encountered during tag data file formatting.
No tag name in message PV: program_name	This message indicates that the OSx mapper was unable to determine the tag name associated with configuration data describing certain internal elements of an FB used in the specified program. This is usually caused by using an out-of-date version of the Engineering Toolset mapper. Make sure the Engineering Toolset mapper is up-to-date as described in the installation instructions (pages 1-4 and following).

D.2 Error Messages for Installation of User-defined Tag Types

Table D-4 shows error messages that may appear when you are installing, deleting, or using user-defined tag types.

Table D-4 Error Messages for User-defined Tag Types

Error Message	Explanation
Cannot add a tag type that already exists in the database.	The tag type already exists in OSx. You can now add instances of that tag type to the system. If you are modifying a user-defined tag type, you must delete it first, then add the modified version as a new tag type.
Failed to add new tag type table to database.	Check the OSx console window for additional information about the nature of the error. Call the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421. In other countries, call 49-911-895-7000.
Cannot delete tag type. Not found in database.	The tag type does not exist in the OSx database. No action is necessary.
Error locating tag type information in database.	Call the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421. In other countries, call 49-911-895-7000.
Cannot delete a user-defined tag type that has configured tags.	You must delete all configured tags of this type from OSx before you can delete the tag type.
Failed to remove tag type table from the database.	Call the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421. In other countries, call 49-911-895-7000.
Operation failed. Missing tag type directory or tag type files.	Call the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421. In other countries, call 49-911-895-7000.
Cannot change to Resize state.	Verify that the system is in Offline state before you attempt to add or remove a user-defined tag type. If the OSx station is part of a multiple-station system, the station must be the sysadmin.
Operation failed. Restoring previous database and returning to OFFLINE.	Check the OSx console window for additional information about the nature of the error. Call the Siemens Energy & Automation, Inc., Technical Services Group in the U.S.A. at 800-333-7421. In other countries, call 49-911-895-7000.

D.3 Miscellaneous Troubleshooting

Global Data

Global data blocks cannot be mapped into OSx. Use the OSx Tag Configurator utility to assign tags to memory addresses in user-created STEP 7 global data blocks. See the chapter on Configuring Tags in the *SIMATIC PCS 7 OSx Process Configuration Manual*.

Naming Restrictions

The space, the character combinations -> and <- and the characters ; , \ ” are invalid for the names of function blocks, chart containers, CFCs, and controller programs. Use of invalid characters makes it impossible to create valid tag installation data for OSx from the S7 mapper file.

Configuring AO, CALC, and IVAR Tags as Symbols

When you configure AO, CALC, or IVAR tags as symbols using the Engineering Toolset, the default values for high_range and low_range are not valid for the OSx database. These default values cause an error on the OSx station when you try to install these tags, and no tags are installed. When you configure AO, CALC, and IVAR tags as symbols, set the upper and lower limits according to the instructions in Section 2.3 under “Configuring Tag Parameters in WinCC Attributes.” Enter the limit values that are to be used on the OSx station.

Index

Customer Response

We would like to know what you think about our user manuals so that we can serve you better. How would you rate the quality of our manuals?

	Excellent	Good	Fair	Poor
Accuracy	_______	_______	_______	_______
Organization	_______	_______	_______	_______
Clarity	_______	_______	_______	_______
Completeness	_______	_______	_______	_______
Graphics	_______	_______	_______	_______
Examples	_______	_______	_______	_______
Overall design	_______	_______	_______	_______
Size	_______	_______	_______	_______
Index	_______	_______	_______	_______

Would you be interested in giving us more detailed comments about our manuals?

☐ **Yes!** Please send me a questionnaire.

☐ **No.** Thanks anyway.

Your Name:	________________________________
Title:	________________________________
Telephone Number:	(_____)___________________________
Company Name:	________________________________
Company Address:	________________________________
	________________________________
	________________________________

Manual Name:	SIMATIC PCS 7 OSx Interface to S7 Controllers	**Edition:**	Original
Manual Assembly Number:	2811155-0001	**Date:**	7/02
Order Number:	6ES7 6550XX058BC8		

FOLD

SIEMENS ENERGY & AUTOMATION INC
3000 BILL GARLAND ROAD
P O BOX 1255
JOHNSON CITY TN 37605–1255

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.3 JOHNSON CITY, TN

POSTAGE WILL BE PAID BY ADDRESSEE

ATTN TECHNICAL COMMUNICATIONS M/S 519
SIEMENS ENERGY & AUTOMATION INC
P O BOX 1255
JOHNSON CITY TN 37605–1255

FOLD