

SIEMENS

SIMATIC

Windows Automation Center WinAC Basis V4.1

User Manual

Version: 05/2004

A5E00340325-01

Copyright and Safety Notification

This manual contains notices that you should observe to ensure your own personal safety, as well as to protect the product and connected equipment. These notices are highlighted in the manual by a warning triangle and are marked as follows according to the level of danger:



Danger

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



Warning

Indicates a potentially hazardous situation that, if not avoided, could result in death or severe injury.



Caution

Used with the safety alert symbol indicates a potentially hazardous situation that, if not avoided, may result in minor or moderate injury.

Caution

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

Notice

Used without the safety alert symbol indicates a potential situation that, if not avoided, may result in an undesirable result or state.

Qualified Personnel

The device/system may only be set up and operated in conjunction with this manual. Only qualified personnel should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground, and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Correct Usage

Note the following:



Caution

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

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

Trademarks

Siemens® and SIMATIC® are registered trademarks of SIEMENS AG.

STEP 7™ and S7™ are trademarks of SIEMENS AG.

Microsoft®, Windows®, Windows XP Professional®, Windows 2000®, and Internet Explorer® are registered trademarks of Microsoft Corporation.

Adobe® and Acrobat® are registered trademarks of Adobe Systems Incorporated.

Copyright Siemens Energy & Automation, Inc., 2004

All rights reserved

The reproduction, transmission or use of this document or its contents is not permitted without express written authority. Offenders will be liable for damages. All rights, including rights created by patent grant or registration of a utility model or design, are reserved.

Siemens AG
Bereich Automatisierungs- und Antriebstechnik
Geschäftsgebiet Industrie-Automatisierungssysteme
Postfach 4848, D-90327 Nuernberg

Disclaimer of Liability

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

© Siemens AG 2004

Technical data subject to change.

Preface

The Windows Logic Controller (WinLC) is the programmable logic controller (PLC) for the Windows Automation Center (WinAC Basis) package. It is fully code-compatible with the SIMATIC product family. You can use many of the SIMATIC products, such as WinCC, with WinLC.

WinLC uses a PROFIBUS-DP network to control the distributed I/O, such as an ET 200M device. WinLC uses S7 communications (such as PROFIBUS, Industrial Ethernet, or MPI) for connecting to STEP 7 or other programming software on another computer.

Audience

This manual is intended for engineers, programmers, and maintenance personnel who have a general knowledge of programmable logic controllers.

Scope

This document describes the features and the operation of WinAC version 4.1.

Other Manuals


You can find additional information in the online help for STEP 7 and in the following manuals:

- *Programming with STEP 7 Manual.* This manual provides basic information on designing and programming a WinLC control program. Use this manual when creating a control program with the STEP 7 automation software.
- *System Software for S7-300/400 System and Standard Functions Reference Manual.* WinLC includes integrated system functions and organization blocks, which you can use when programming. This manual provides you with descriptions of the system functions, organization blocks, and loadable standard functions.
- *Working with STEP 7 Getting Started Manual.* This manual explains the usage and the functions of the STEP 7 automation software. This manual provides you with an overview of the procedures used to configure WinLC and to develop control programs.

To find these and other manuals, select the **Start > Simatic > Documentation** menu command from the Start menu of the computer where STEP 7 and SIMATIC NET are installed.

Additional Assistance


For assistance in answering technical questions, for training on this product, or for ordering, contact your Siemens distributor or sales office.

 North America and South America

Telephone: +1 (800) 333-7421

Fax: +1 (423) 262-2200

simatic.hotline@siemens.com

 Europe and Africa

Telephone: +49 (0) 180 5050 222

Fax: +49 (0) 180 5050 223

adsupport@siemens.com

 Asia and Pacific region

Telephone: +86 10 64 75 75 75

Fax: +86 10 64 74 74 74

adsupport.asia@siemens.com

Contents

Product Overview	1
Introduction to PC-Based Control	1
Introduction to the Control Panel	1
WinLC Features.....	2
SIMATIC Functionality Supported by WinLC.....	2
Windows Functionality Supported by WinLC.....	2
New Features for WinAC V4.1.....	3
System Requirements	5
Windows User Privileges.....	6
Using Help	7
Accessing Help from the Control Panel.....	7
Using the Table of Contents	7
Using the Index.....	8
Using Full-Text Search	8
Printing Help Topics.....	8
Changing the Language of a Help Topic.....	8
Installation	9
Overview of the Installation Tasks	9
Installing SIMATIC NET and the CP Card	9
Installing the WinAC Software	10
Authorizing the WinAC Software	11
Installing the Authorization for the First Time	11
Adding an Authorization at a Later Date.....	11
Removing an Authorization.....	12
Running the WinLC Controller without the Authorization	12
Recovering the Authorization in Case of a Defective Hard Drive.....	12
Uninstalling the WinAC Software.....	12
Getting Started	13
Getting Started Overview.....	13
Prerequisites	13
Understanding the Concepts	14
What Is a PC Station?.....	14
What Is an Index?	16
What Is a Submodule?.....	17
What Is an IF Slot?	19
Configuring the CP Card as a Submodule.....	20
Use Case: Designating a CP Card as a Submodule	20
Testing the Submodule CP Card	22
Configuring the Controller in STEP 7.....	24
Use Case: Connecting STEP 7 to WinLC.....	24
Use Case: Hardware Configuration in STEP 7.....	26
Invalid Characters	28

Controller Operations	29
Starting and Shutting Down the Controller	29
Starting WinLC	29
Shutting Down WinLC	29
Closing the Control Panel	29
Changing the Operating Mode of the Controller	30
Relating the Mode Buttons and Status Indicators to the Operating Mode of the Controller...	30
Using the Mode Buttons to Control Access to the Controller	31
Resetting the Memory Areas: MRES Command (CPU Menu)	32
Using the Status Indicators	33
Corrective Action If the STOP Indicator is Flashing Slowly	34
Corrective Action If All Status Indicators Are Flashing	34
Using the Tuning Panel	35
Using the Diagnostic Buffer	37
Sorting Events (upper panel)	38
Choosing Format (lower panel)	38
Saving the Diagnostic Buffer	38
Displaying Help	38
Archiving and Restoring Control Programs	39
Creating an Archive File	39
Restoring an Archive File	39
Storing Retentive Data	40
Storing Information About the Controller	40
Restoring Memory Areas on Start Up	42
Using SFCs to Retain Data	44
Uninterruptible Power Supply (UPS)	44
Customizing and Security Options	45
Customizing Options	45
Selecting the Language	45
Selecting the Autostart Feature	45
Setting the Security Options	46
Changing the Password	47
Startup Options for the Controller	48
Starting the Controller at PC Boot	48
Setting the Restart Method	49
STEP 7 Operations	51
Using STEP 7 with the Controller	51
Accessing WinLC from STEP 7	51
Configuring the Operational Parameters	52
Configuring the Operational Parameters for WinLC	52
Configuring the Startup Characteristics	54
Configuring the Clock Memory	55
Configuring the Scan Cycle	56
Configuring Retentive Memory Areas	58
Configuring the Time-of-Day Interrupt	59

Configuring the Interrupts	60
Configuring a Cyclic Interrupt	61
Configuring the I/O Addresses.....	62
Assigning Addresses for the DP I/O	62
Accessing the Data in the DP I/O Modules.....	65
Specifying the Diagnostic Addresses for the DP I/O Modules.....	68
STEP 7 Components.....	69
Logic Blocks Supported by WinLC	69
Peer-to-Peer Communication Functions.....	70
PROFIBUS DPV1	71
Organization Blocks (OBs).....	72
System Functions (SFCs).....	75
System Function Blocks (SFBs).....	80
System Clock and Run-Time Meter.....	82
Tuning the Performance of the Controller	83
Scan Cycle for a PC-Based Controller	83
Tasks Performed during the Scan Cycle	83
Methods for Managing the Performance of WinLC	85
What Causes Jitter?	86
Priority Settings for Different Applications Can Cause Jitter	86
Priorities among the WinLC Threads Can Cause Jitter.....	88
The Sleep Interval Forced by the Execution Monitor Can Cause Jitter.....	89
Adjusting the Priority of the Controller	90
Windows Priorities	91
Background Priority (1 to 7).....	92
Managing the Sleep Time.....	93
Sleep Management Techniques	93
Adjusting the Minimum Sleep Time and Cycle Time	97
Using SFC47 to Add Sleep Time in the Control Program	100
Adjusting the Sleep-Monitoring Algorithm of the Execution Monitor	101
Example: Avoiding Jitter in the Start Time of an OB	107
Advanced Topics.....	111
Connecting WinLC to the SIMATIC NET OPC Server.....	111
Task Overview	111
Step 1: Add the OPC Server to the PC Station	112
Step 2: Configure WinLC for Using the OPC Server	114
Step 3: Add the OPC Connection to the WinLC Configuration	117
Step 4: Download the configuration to the controller.....	121
Step 5: Connect WinLC to the OPC Server.....	122
Reference Information	129
Technical Data.....	129
Order Number	129
Features.....	129
Technical Specifications	129
Execution Times	132

Contents

Execution Times of Instructions.....	132
Execution Time of DP Instructions.....	134
Performance Test: Scan Time Jitter	135
Results of the Performance Test	136
Troubleshooting.....	137
Troubleshooting Network Problems.....	137
Responding to Diagnostic Events.....	138
System Status List (SSL).....	139
Using SFC51 to Read the SSL	139
SSL_ID Descriptions.....	140
Glossary.....	151

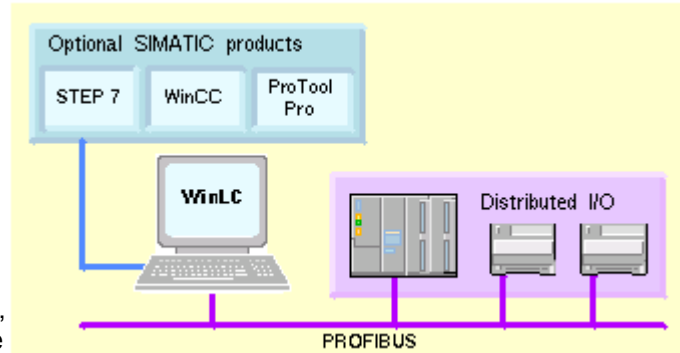
Product Overview

Introduction to PC-Based Control

WinLC is a programmable "soft logic" controller — a software version of an S7 controller — that runs on a standard computer (PC).

WinLC supports multiple networks and connects to the distributed I/O, such as ET 200M, by means of CP cards (CP 5611, CP 5613 V3, or CP 5613 V6 or later) that you install in your computer.

As part of the SIMATIC family of automation products, WinLC can also communicate with STEP 7 or other SIMATIC products, such as WinCC, ProTool Pro, or other SIMATIC S7 controllers, including WinLC, over PROFIBUS, Ethernet, or MPI networks. This allows you to use WinLC in a typical factory automation configuration.




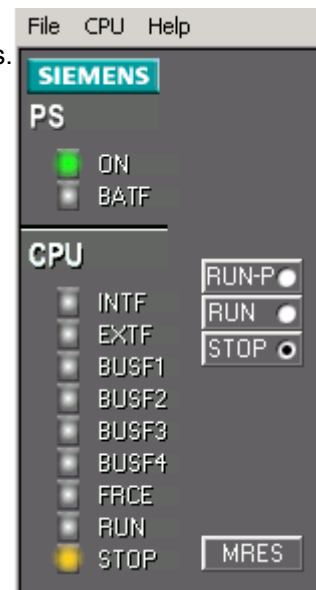
Introduction to the Control Panel

The control panel corresponds to the faceplate of the S7 controllers. It enables you to start or shut down the controller and to perform other controller operations.

The control panel contains the following elements for working with the controller:

- Three mode buttons for changing the operating mode of the controller (The mode buttons are similar to the keyswitch on a hardware S7 controller)
- Status indicators for the controller
- An MRES button for resetting the memory areas
- Menus for controller operations

An icon  is displayed in the Windows taskbar whenever the controller is operating. When the controller is operating and the control panel is closed, you can double-click this icon to open the control panel. Opening or closing the control panel does not influence the state of the controller. The status of the operator switches and the LEDs are stored in the controller.












WinLC Features

WinLC is a software version of an S7 controller that executes on a Windows PC. It executes S7 control programs like other S7 controllers and allows for easy integration with STEP 7 and standard Windows applications.

SIMATIC Functionality Supported by WinLC

WinLC provides the following features:

-  Implements a comprehensive subset of the S7 logic blocks of SIMATIC controllers: Organization Block (OB), System Function Block (SFB), and System Function (SFC)
-  Uses PROFIBUS-DP as its I/O subsystem, supporting DPV0 and DPV1 slaves (PROFIBUS DPV1 provides enhanced alarm and status reporting, in order to communicate with intelligent slave devices)
-  Uses S7 communication services, offering compatibility with SIMATIC applications such as STEP 7 and WinCC (for tasks such as programming, debugging, or monitoring)
-  Supports up to 4 separate PROFIBUS-DP subnets for controlling distributed I/O
-  Supports the routing of S7 communications through the submodule CP cards of WinLC, allowing STEP 7 on one subnet to connect to an S7 station (such as an S7-400 controller) on a different subnet
-  Provides ability to archive and restore control programs
-  Allows you to control the operational mode of WinLC and to view status information from the control panel
-  Provides a tuning panel for optimizing system performance
-  Allows peer-to-peer communications between controllers (hardware or software) on the network


Windows Functionality Supported by WinLC



Windows Administrator privileges (ADMIN) are not required in order to operate the WinLC controller. With Power User, User or even with Guest privileges, you can perform operational tasks, such as changing the operating mode of the controller from RUN to STOP, modifying the sleep time, or restoring an archived control program.

If you configured WinLC to start at PC boot (and if the controller was consequently started by rebooting the computer), one user can log off and another user can log on without affecting the operation of the controller.


Note: Although WinLC supports logging off and logging on as a Windows user, the Windows XP "Switch User" function is **not** supported by WinLC.


New Features for WinAC V4.1

-  **Improved performance** Control programs running on WinLC V4.1 have execution times up to twice as fast as the same control program running on WinLC V4.0. The reduction in scan time allows the PC resources to be utilized by other tasks, such as HMI or data processing applications.


Other enhancements in WinLC V4.1 reduce the effect of the STEP 7 programming tools, such as block status and breakpoints, on the performance of the WinLC controller. In addition, WinLC V4.1 includes support for new functionality and improved interactions with STEP 7 and PCS 7
-  **Peer-to-peer communication** Using STEP 7 V5.3 and a CP card that supports data exchange with broadcast, WinAC V4.1 provides peer-to-peer communications between DP slave devices.
-  **Multi-PLC support** Up to 3 WinAC Slot PLCs (V3.4 or higher) can run on the same computer with WinLC V4.1 controller.

While the PLCs can communicate with each other (using S7 communication commands such as BSEND and BRCV), each of the PLCs operates independently. The functionality of the multiple PLCs is comparable to the operation of multiple S7-400 CPU modules in a segmented rack.

All of the multiple PLCs can be connected simultaneously to SIMATIC ProTool/Pro, WinCC, or SIMATIC NET OPC application.
-  **SIMATIC NET OPC server** WinAC V4.1 uses the OPC server of SIMATIC NET for connecting to OPC client applications. Using the SIMATIC NET OPC server improves the performance of OPC connections (compared to the OPC server of SIMATIC Computing).


The SIMATIC NET OPC server provides better integration into STEP 7 configuration, allowing you to use STEP 7 symbols when configuring your OPC connections.
-  **Windows Administrator privileges not required** You are not required to have Windows Administrator privileges in order to operate the WinAC V4.1 controller.

With Power User, User or even with Guest privileges, you can change the operating mode of the controller, modify the sleep time or the minimum scan time, archive or restore a control program, or set the security option, in addition to other operational tasks.


-  **WinAC ODK V4.1 interfaces**

You can use the interfaces of the WinAC Open Development Kit (ODK) to implement custom applications into your control tasks:

 - Custom Code Extension (CCX): allows your control program to call custom DLLs directly from the control program being executed by the PLC
 - Shared Memory Extension (SMX): enables a very fast and efficient data exchange between the PLC and an application by providing direct access to a specified PI/PO area (utilizing dual-port RAM)
 - Controller Management Interface (CMI): allows you to embed the functionality of the WinAC control panel into a custom application


-  **CP 5613-A2**

WinAC V4.1 supports the CP 5613-A2 card.


-  **Non-retentive DBs**

Using STEP 7 V5.3, you can configure a non-retentive data block (DB) for your control program.

SFC82 and SFC85 also support the creation of non-retentive DBs.

-  **Diagnostic tool for computer-related problems**

WinAC V4.1 provides a software application that provides system-level information for the customer service engineers to use in diagnosing computer-related problems, such as when Windows crashes.

-  **Enhanced support for Positioning and Motion applications**

Using a distributed configuration (PROFIBUS-DP and ET 200M), WinAC V4.1 supports the following SIMATIC S7 Function Modules for simple point-to-point positioning and for complex motion profiles:

 - FM 353: Positioning module for stepper motors in machines with high clock-pulse rates
 - FM 354: Positioning module for servo motors in machines with high clock-pulse rates

WinAC V4.1 also supports the use of SIMOVERT MASTERDRIVES with complex motion functions, such as CAM control or electronic gear.

System Requirements

To use WinAC, your personal computer (PC) must meet the following criteria:

Operating System	Microsoft Windows 2000 Professional Service Pack 3 or Microsoft Windows XP Professional Service Pack 1
Processor and memory	<p>Pentium uniprocessor or multiprocessor system:</p> <ul style="list-style-type: none"> • 800 MHz or faster, single or dual processor • 256 Mbytes or more of RAM • BIOS must support plug-and-play
Hard drive	<p>A hard drive with 125 Mbytes of free space</p> <p>The setup program uses at least 1 Mbyte additional free space on drive C for the WinLC Setup program (Setup files are deleted when the installation is complete)</p>
Operator interface	A color monitor, keyboard, and mouse or other pointing device (optional) that are supported by Windows
Network interface	At least one PCI slot with an installed Siemens CP card (CP 5611, CP 5613 V3, or CP 5613 V6 or later) connected to a PROFIBUS-DP network for distributed I/O communication
Siemens software	<p>Communications software: SIMATIC NET V6.2</p> <p>Note: The SIMATIC NET software must be installed before you install WinAC.</p> <p>Programming and configuration software: STEP 7 V5.3 or later</p> <p>Note: You can use STEP 7 V5.2 if you do not need any of the following WinAC 4.1 new features:</p> <ul style="list-style-type: none"> • Improved performance • Peer-to-peer communication between DP slave devices • Enhanced support for positioning and motion applications • Non-retentive DBs • Multi-PLC support • WinAC ODK 4.1 • SSL_ID 0x1C

Windows User Privileges

You are not required to have Windows Administrator (ADMIN) privileges in order to perform WinAC operations, such as changing the operating mode of the controller, modifying the sleep time or the minimum scan time of the controller, archiving or restoring control programs, or setting the security options.

With Power User, User or even with Guest privileges, you can perform any operation from the WinLC control panel. This allows you to manage the network privileges for the PC station within your application and to avoid conflicts during installation, commissioning and operation of a PC-based automation solution that is part of a larger system.

As shown in the following table, some operations are restricted to certain Windows User privilege classes.

Operation	Administrator	Power User	User	Guest
Installing WinAC software	<input checked="" type="checkbox"/> Allowed	Not allowed	Not allowed	Not allowed
Configuring or modifying the PC Station	<input checked="" type="checkbox"/> Allowed	<input checked="" type="checkbox"/> Allowed	<input checked="" type="checkbox"/> Allowed	Not allowed
Performing WinAC operations	<input checked="" type="checkbox"/> Allowed	<input checked="" type="checkbox"/> Allowed	<input checked="" type="checkbox"/> Allowed	<input checked="" type="checkbox"/> Allowed

Using Help

The online help system provides information about the control panel and the controller. This topic provides information about using online help:

- Accessing Help from the Control Panel
- Using the Table of Contents
- Using the Index
- Using Full-Text Search
- Printing Help Topics
- Changing the Language of a Help Topic


Accessing Help from the Control Panel

To access online help from the control panel, use one of the following methods:


- You can click one of the following entries on the Help menu to display the electronic manual that provides the online help:

 **Help on Panel**

The **Help > Help on Panel** command opens the initial page of the online help for the control panel when it is not connected to a controller. The table of contents is in the left navigation pane.

 **Help on Controller** (available when panel is connected to a controller)

The **Help > Help on Controller** command displays the initial page of the online help for the controller that is connected to the control panel. It describes controller and control panel operations. The table of contents is in the left navigation pane.

 **Introduction** (available when panel is connected to a controller)

The **Help > Introduction** command displays a topic that provides an introduction to the controller and the tasks you can perform with it.

 **Getting Started** (available when panel is connected to a controller)

The **Help > Getting Started** command displays a topic that helps you get started when you begin using the control panel to work with the controller for the first time.

- You can click the Help button in a dialog or message box to view information about that specific dialog or message box.
- You can press the F1 key to view context-sensitive help on the currently selected item (for example, a window, dialog, or menu).

Using the Table of Contents

The table of contents is in the left navigation pane of the browser; you can navigate to any topic by clicking the mouse:

- Click a book to open it and display the books and topics that it contains.
- Click the book again to close it.
- Click any topic within the table of contents to display that topic.

The topic you are currently viewing is highlighted in the table of contents.

The table of contents can be either hidden or displayed:

- Click the "x" in the left navigation pane to close the table of contents.
- Select the Contents tab or the "Show" link in a topic to display it. (The "Show" link appears only when you have displayed a context-sensitive topic from the application.)

Using the Index

You can use the index to find information about a specific subject. Use one of the following methods to access the index:

- Select the Index tab in the navigation pane. (If the Index tab is not visible, click the "Show" link at the top of the topic. The "Show" link appears only when you have displayed a context-sensitive topic from the application.)
- Click the Index button in any help topic.

Using Full-Text Search

The help system also provides full-text search capabilities. You can use the search field that is displayed above the topic, or select the Search tab from the navigation pane. (If the Search field and Search tab are not visible, click the "Show" link at the top of the topic. The "Show" link appears only when you have displayed a context-sensitive topic from the application.)

You can use the Boolean operators AND, OR, and NOT and parentheses in your search expression. Wildcards ("*") are not supported.

Printing Help Topics

To print a single topic that is displayed in your browser, right-click in the topic pane and select Print from the context menu. Select the print options of your choice.

Changing the Language of a Help Topic

The browser contains language buttons for each of the supported languages. To see the current help topic in another language, click the language button of your choice. The current topic is displayed in the language you selected, but the contents, index, and search features of the online help system remain in the original language. This may be helpful if a topic is unclear and you want to read it in another language.

If you select a language that you did not install, the online help system cannot display the topic in that language and displays a "Page not found" error. Changing the language of an online help topic does not change the display language of the control panel.

Installation

Overview of the Installation Tasks

You must install the following products in order to complete the installation of WinAC:

- Communications processor (CP) card and SIMATIC NET software
- WinAC software
- WinAC Authorization

Note: You must have Windows administrator (ADMIN) privileges to install the WinAC Basis software.

You must install the SIMATIC NET software before installing the WinAC components.

To install WinAC, you must complete the following tasks:

- Install the SIMATIC NET software.
- Install the WinAC software.
- Authorize the WinAC software.

Installing SIMATIC NET and the CP Card

WinAC supports the following Siemens CP cards for communicating with the distributed I/O over a PROFIBUS-DP subnet: CP 5611, CP 5613 V3, or CP 5613 V6 or later.

You can install up to four CP cards in your computer. For specific installation instructions, refer to the documentation for your CP card.

WinAC also requires that you install the SIMATIC NET V6.2 software.

To use a CP card with WinAC, you must install it as a submodule.

Notice

Assigning the CP card as a submodule of WinLC automatically changes the operation mode of the CP card to PG Operation. However, the Set PG/PC Interface dialog allows you to change the Interface Parameter Assignment for the CP card when the WinLC controller is not running. Using the Set PG/PC Interface dialog to change the Interface Parameter Assignment of a CP card that has already been configured as a submodule causes an error that prohibits WinLC from going to RUN mode.

For information about recovering from this error, refer to the Use Case on designating the CP card as a submodule.

Installing the WinAC Software

The WinAC software includes a Setup program that guides you step by step through the installation procedure. The Setup program begins automatically when you put the CD in the drive. If the Setup program does not start automatically, double-click the Setup.exe file on the CD.

Note: You must have Windows administrator (ADMIN) privileges to install the WinAC software.

If you select the "Run automatic authorization" checkbox in the Target Drive dialog, the Setup program checks to see whether an authorization is installed on the hard disk. If no authorization is found, a message notifies you that the software can be used only with an authorization. You can run the authorization program immediately or you can continue the installation and execute the authorization later.

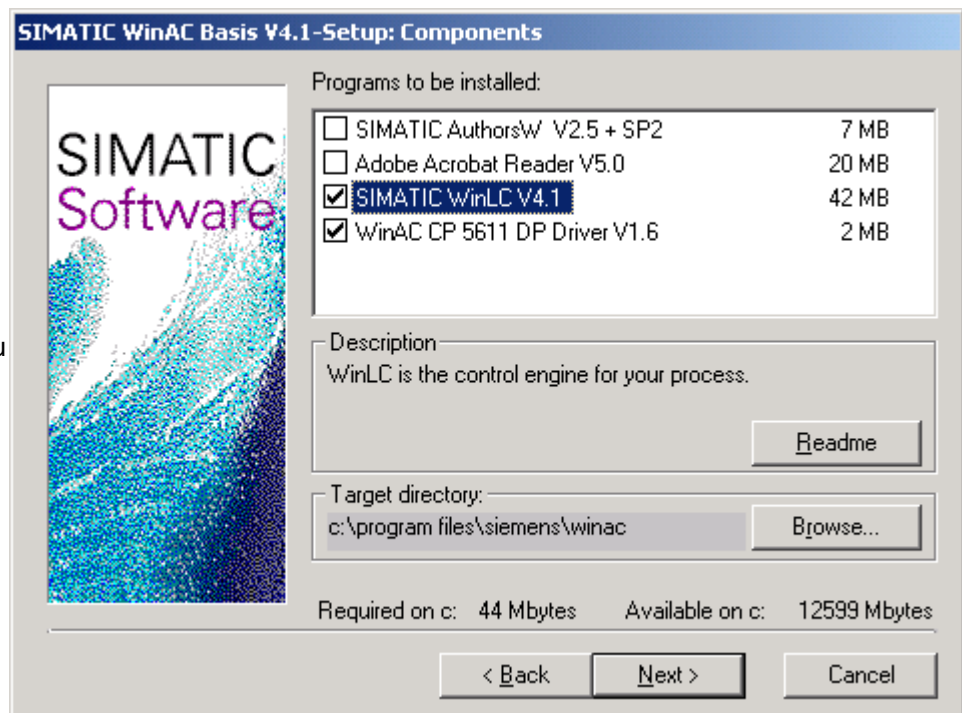
Follow the instructions of the Setup wizard. From the list of WinAC components, select the components to be installed.

Note: To use a CP 5611 card or the integrated PROFIBUS interface of a SIMATIC PC as a submodule of WinLC, you must install the WinAC DP driver for the CP 5611.

If you get messages from SIMATIC NET that say the CP card is configured for use with SIMATIC NET and STEP 7, click OK. This is a normal part of the installation process.

After installing the components WinAC, the Setup wizard prompts you to insert the WinAC authorization diskette in drive A.

The Setup program notifies you that the installation is complete.



Authorizing the WinAC Software

The WinAC software requires a product-specific authorization (or license for use). There are separate authorization diskettes for each of the SIMATIC automation software products. You must install the authorization for each product.

Caution

If improperly transferred or removed, the authorization for the WinAC software may be irretrievably lost.

The Readme file on the authorization diskette contains guidelines for installing, transferring, and removing the authorization for the WinAC software. If you do not follow these guidelines, the authorization for the WinAC software may be irretrievably lost.

Read the information in the Readme file on the authorization diskette, and follow the guidelines in regard to transferring and removing the authorization.

Installing the Authorization for the First Time

When you install the software for the first time, a message prompts you to install the authorization. Use the following steps to install the authorization for the WinAC software:

1. When prompted, insert the authorization diskette in a drive.
2. Acknowledge the prompt.

The authorization is transferred to a physical drive, and your computer registers the fact that the authorization has been installed.

Adding an Authorization at a Later Date

If you attempt to start the WinAC software and no authorization is found, a prompt appears on the screen. If you want to install the authorization, you can do so using the AuthorsW program. The AuthorsW program allows you to display, install, and remove authorizations. To authorize the WinAC software, follow these steps:

1. Select the **Start > Simatic > AuthorsW > AuthorsW** menu command.
2. Select the Move Authorization tab from the AuthorsW dialog to display the authorizations available.
3. Select the authorization for WinAC 4.1 from the list of available products on the authorization diskette.
4. Click the right arrow to transfer the authorization from the diskette to your hard drive.

If AuthorsW is not installed on your computer, insert the WinAC installation CD. When the Setup Components dialog appears, select the AuthorsW checkbox. After the installation is complete, select the AuthorsW program from the **Start** menu as described above.

Removing an Authorization

If you need to repeat the authorization (for example, if you want to reformat the drive on which the authorization is located), you must first remove the authorization. You need the original authorization diskette to do this.

Use the following steps to transfer the authorization back to the authorization diskette:

1. Insert the original authorization diskette in the drive.
2. Select the **Start > Simatic > AuthorsW > AuthorsW** menu command.
3. From the list of all authorizations on drive C, select the authorization to be removed.
4. Select the **Authorization > Transfer** menu command.
5. In the dialog, enter the target drive to which the authorization will be transferred.
6. The window with the list of authorizations remaining on the drive is then displayed. Close the AuthorsW program if you do not want to remove any more authorizations.

You can then use the diskette again to install an authorization.

Running the WinLC Controller without the Authorization

If you remove (or accidentally delete) the authorization for the software, the WinLC controller continues to operate; however, a notification message appears every six minutes to alert you that the authorization is missing.

Recovering the Authorization in Case of a Defective Hard Drive

If a fault occurs with the authorization file on your hard disk or on the authorization diskette, contact your local Siemens representative.

Uninstalling the WinAC Software

Use the following procedure to remove the WinAC software from your computer:

1. Double-click the Add/Remove Programs icon in the Windows Control Panel.
2. Select the WinAC component entry in the displayed list of installed software. Click Add/Remove to uninstall the software. Each component of WinAC is listed separately.

If the Remove Shared File dialog boxes appear, click No if you are unsure how to respond.

Getting Started

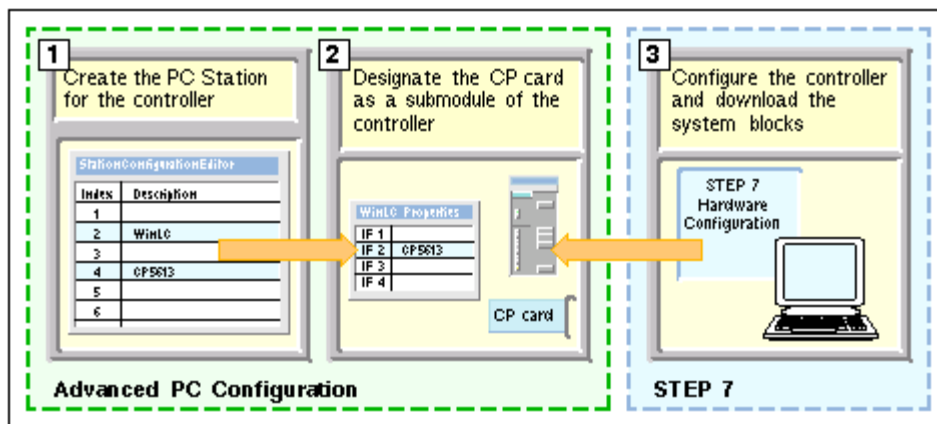
Getting Started Overview

The Getting Started section helps you to establish communications between the WinLC controller, STEP 7, and the distributed (DP) I/O by performing the following tasks:

- Use the Station Configuration Editor to designate the CP card as a submodule of WinLC.
- Use STEP 7 to configure the controller and CP cards and to download the system blocks.

The Getting Started section also helps you understand the basic concepts for setting up WinLC: PC station, Index, Submodule, and Interface (IF) Slot.

To establish communications for the WinLC controller, you must perform the tasks shown in the following figure. (Click on one of the numbered sections to see a use case describing that task.)



Prerequisites

Before you can get started, you must install the following components:

- SIMATIC Communications Processor (CP) card, such as a CP 5613
- SIMATIC NET communications software
- WinLC software
- STEP 7 programming software

Note: STEP 7 can be installed either on the same computer as WinLC or on a remote computer.

Understanding the Concepts

What Is a PC Station?

The PC station functions as a software-based virtual rack for creating a PC-based automation system. You create the PC station in the Station Configuration Editor when you install the SIMATIC NET software. When you install WinLC, it automatically appears in a slot (index) of this virtual rack in the Station Configuration Editor. You also add one or more CP cards to your PC Station, which you subsequently configure as a submodule of WinLC.

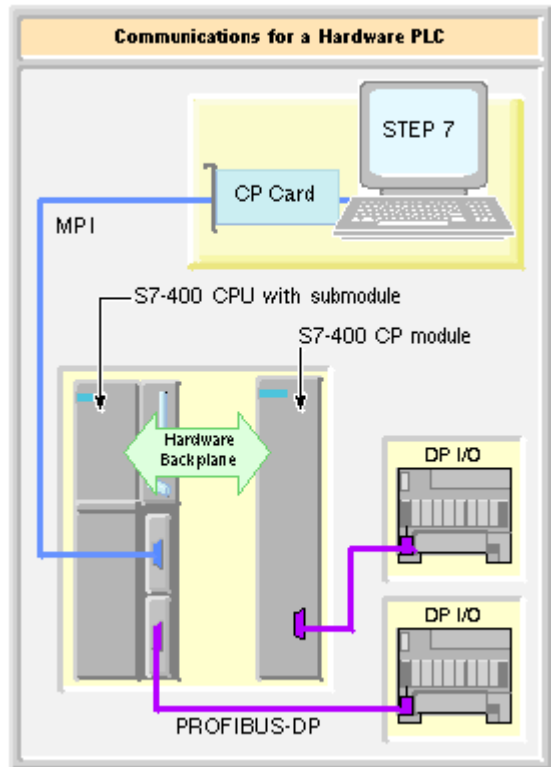
The PC-based controller (WinLC) is similar to an S7-400 hardware controller.

The figure shows a communication model for a hardware controller.

The hardware controller consists of modules that are inserted into a rack. These modules communicate over the backplane of the rack.

- STEP 7 communicates with the controller (in this example, an S7-400 CPU module) over an MPI subnet, using a CP card that is installed in the computer.
- The controller communicates with other modules, such as a CP module) over the backplane of the rack.
- The CP module communicates with the distributed I/O over a PROFIBUS-DP subnet.

For this example, the S7-400 controller has a communications submodule that allows the controller to communicate with distributed I/O over a PROFIBUS-DP subnet.



WinLC communicates very much like the hardware controller.

Because WinLC is located on your computer, WinLC uses one or more CP cards (CP 5611, CP 5613 V3, or CP 5613 V6 or later) for communicating to the distributed (DP) I/O remotely through the PROFIBUS-DP network. You use the Station Configuration Editor to configure the CP cards as submodules.

WinLC and any CP card that is not configured as a submodule communicate over a software "backplane."

If STEP 7 is installed on the same computer as WinLC, STEP 7 communicates over the software "backplane."

WinLC uses a submodule CP card to communicate with the DP I/O.

STEP 7 on a remote computer uses a CP card to communicate with WinLC. In this example, the local CP card uses the software "backplane" to communicate with WinLC.

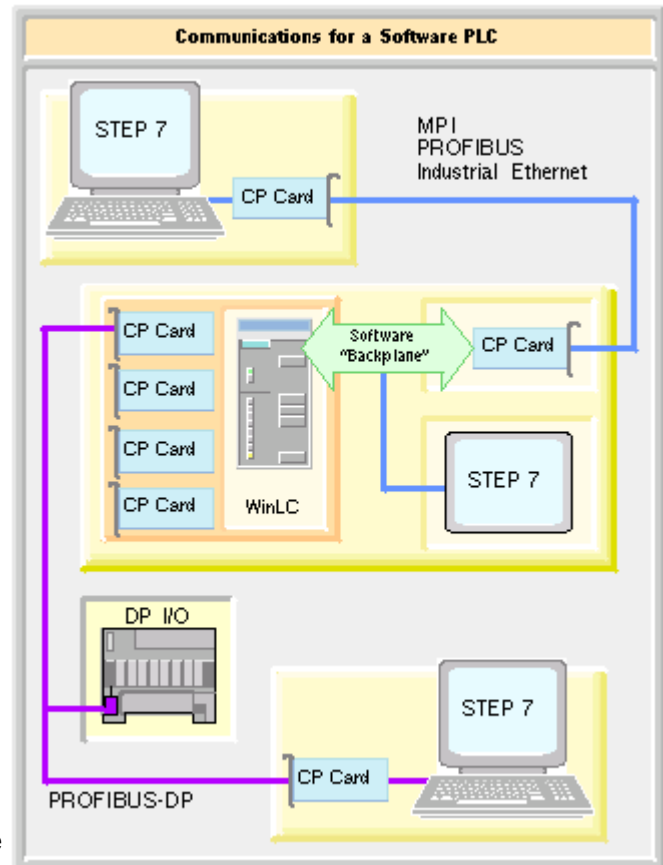
STEP 7 on a PROFIBUS-DP subnet can communicate with WinLC through the submodule CP card.

Configuring the "Hardware" of the PC-Based Controller

In the same way that you use STEP 7 to create the control program and system blocks for a hardware controller, you must use the Hardware Configuration tool of STEP 7 to configure the components that you installed in the PC station. When you download the system blocks to WinLC, the name of the PC station is renamed to the name as configured by STEP 7.

After you configure your PC station both in the Station Configuration Editor and in STEP 7, you can download your control program to the PC station.

Note: To use the CP card for communicating both with STEP 7 and with the distributed I/O may require an additional software license. See your Siemens sales representative or distributor for more information.



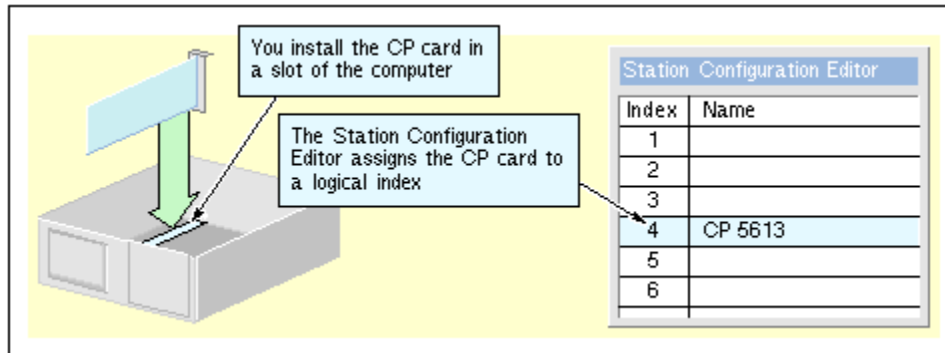
What Is an Index?

An index is a numbered slot in the virtual rack of the PC station. The PC Station provides slots for the SIMATIC components of a PC-based automation solution, including not only WinLC and CP cards, but also SIMATIC HMI and SIMATIC NET OPC.

Each slot in the PC station is assigned a number or index. When you install a SIMATIC component, such as WinLC or the CP card, the component is inserted in an index line in the Station Configuration Editor. The Station Configuration Editor shows the configuration of your PC station.

Note: You install a CP card in a physical location (one of the PCI slots) in your computer. There is no implied or required correspondence between this physical location and the index of the CP card in the Station Configuration Editor.

The index number for a CP card in a PC station is its virtual slot on the virtual rack. This can be any index number you choose. It does not have to be the same as the hardware slot number for the CP card. The index number in the Station Configuration Editor, however, must be the same as the slot number in the STEP 7 Hardware Configuration tool for the same component.



What Is a Submodule?

In order for WinLC to communicate with the DP I/O devices on the PROFIBUS-DP network, you must designate a PROFIBUS CP card to be a submodule for the controller. With this submodule approach, WinLC has full control over the CP card, providing optimum performance and determinism for operating the I/O. WinLC supports up to four submodules.

If you use a CP 5611 card as a submodule of WinLC, note that you can insert only one CP 5611 in the PC Station.

You can configure up to a total of four CP cards as submodules for WinLC. CP 5613 cards can be inserted in any of the four IF slots.

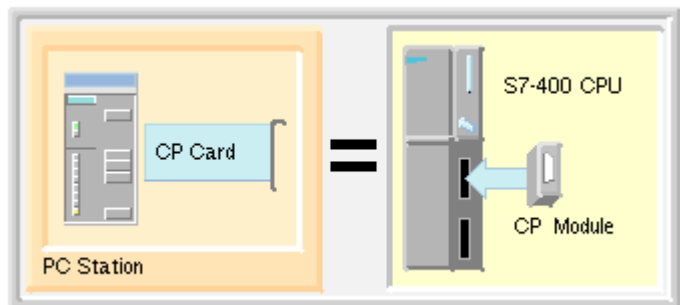
To configure the CP card as a submodule, you move the CP card from an index in the PC station to an IF slot in the controller.

After you have configured the CP card as a component of the PC station, you can use it only for SIMATIC communications (such as MPI, TCP/IP, Industrial Ethernet, or PROFIBUS) with STEP 7, SIMATIC HMI, or other SIMATIC controllers. For example, you can download a program from STEP 7 to WinLC. However, the CP card cannot communicate with the distributed (DP) I/O on the PROFIBUS-DP network until you designate it as a submodule of the controller.

Configuring a CP card as a submodule of WinLC is like installing a CP submodule into a slot of an S7-400 CPU.

Designating the CP card as a submodule of WinLC not only allows WinLC to use that CP card for SIMATIC communications when WinLC is active, but also allows WinLC to use that CP card for communicating with the DP I/O.

In order for a submodule CP card to be used for SIMATIC communications with an application other than WinLC (on the PC station), the second application must be a configured part of the PC station.



Notice

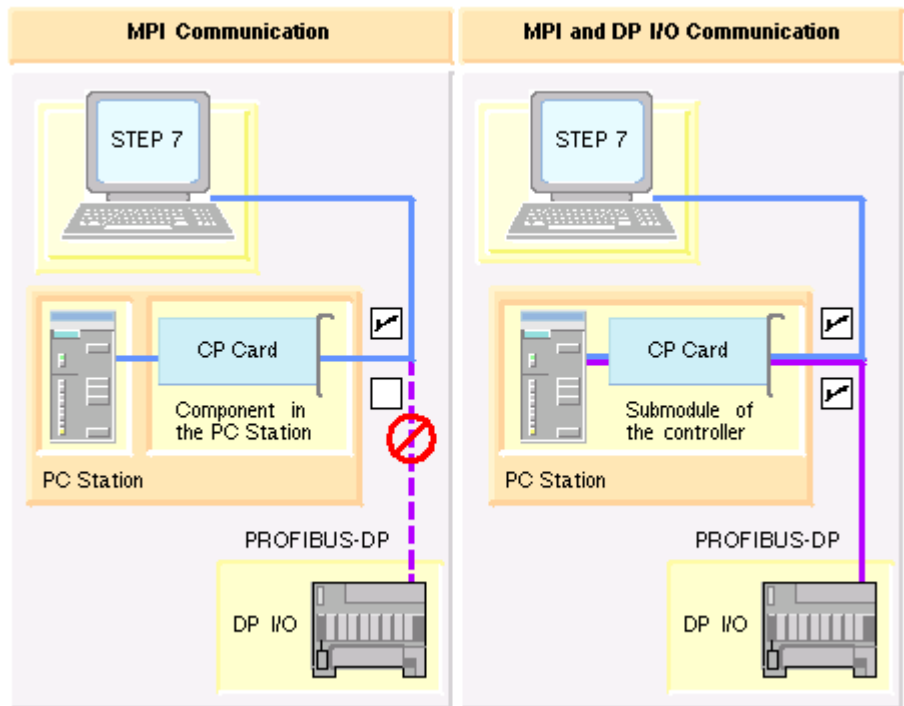
Any change to the configuration of a submodule causes WinLC to delete the control program. You must then download a new control program and configuration.

For example: Moving the submodule CP card to a different slot, WinLC recognizes that the hardware configuration has changed and deletes the control program, even if you had reconfigured the CP card for the same IF slot in the configuration of the PC station. You must still reload the control program from STEP 7.

The figure shows the difference between the CP card as a component of the PC station and the CP card as a submodule of WinLC.

With the CP card configured as a component of the PC station, WinLC can communicate with STEP 7 on a remote computer, but cannot communicate with the DP I/O.

With the CP card configured as a submodule, WinLC can communicate with both STEP 7 on a remote computer (using SIMATIC communications) and with the DP I/O on the PROFIBUS-DP subnet.



What Is an IF Slot?

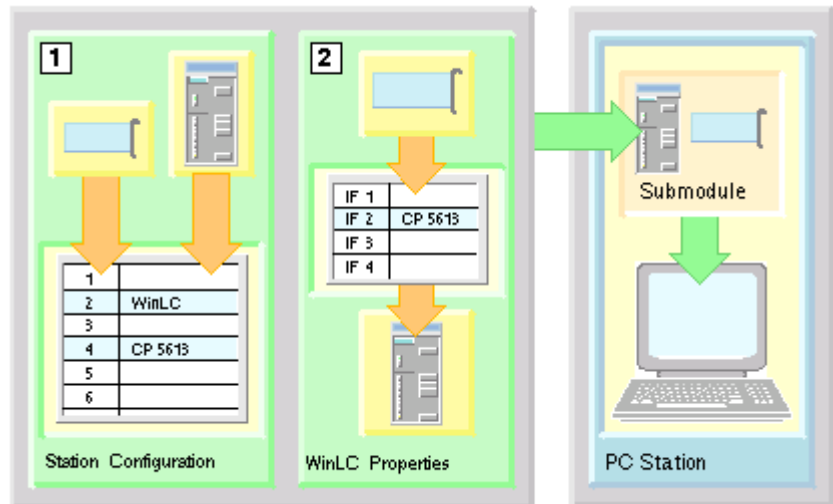
WinLC provides four interface (IF) slots for designating CP cards as submodules. WinLC must have exclusive control of any CP card that you move to an IF slot. This allows the controller to communicate with the distributed (DP) I/O.

To use the DP network to communicate with I/O, you must configure the CP card to be a submodule of WinLC. You use the WinLC Properties dialog to move the CP card to one of four interface slots, IF1 through IF4.

The IF slot number of the submodule is independent of the PCI hardware slot for the CP card. However, the IF slot number for the CP card submodule in the Station Configuration Editor must match the IF slot number in the STEP 7 Hardware Configuration tool.

You first install both the CP card and WinLC in the PC station. You then move the CP card into one of the IF slots for WinLC.

For more information, refer to the following topic: Use Case: Designating a CP Card as a Submodule



Configuring the CP Card as a Submodule

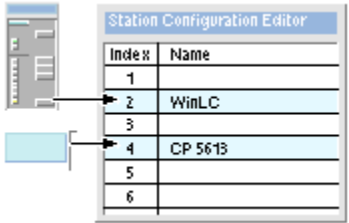

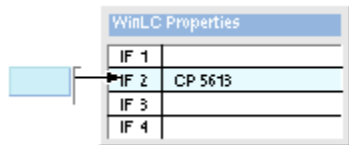
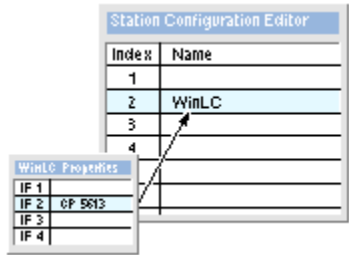
Use Case: Designating a CP Card as a Submodule

Goal: To configure a CP card as a submodule of WinLC by inserting it into an IF slot of the controller, and to verify that configuration. The CP card configuration enables WinLC to communicate with DP I/O over the PROFIBUS network.

Software Tool: You use the Station Configuration Editor of SIMATIC NET to move the CP card from the virtual rack to the submodule slot of WinLC.

Next Step: You configure WinLC and all other components of the PC station in STEP 7.

To configure your CP card as a submodule, ensure that the controller is shut down and follow these steps:

	<ol style="list-style-type: none"> 1. Double-click the computer icon  on the Windows taskbar. The Station Configuration Editor appears with WinLC in the second index. 2. Select an empty index in the PC Station and click the Add button to display the Add Component dialog. 3. From the drop-down list, select the CP card to add to the station configuration and click OK. The Component Properties dialog appears. You can use this dialog to make changes to component properties such as baud rate if necessary. Click OK to add the component to the station. 4. Select WinLC and click the Edit button to display the Edit Component dialog. You can use this dialog to change the index number for the WinLC controller. 5. Click the Properties button to display the WinLC Properties dialog.
	<p>In the SubModule tab, the four submodule interfaces are listed (IF1 to IF4).</p> <ol style="list-style-type: none"> 1. Click an empty slot to highlight the IF slot. 2. Click the Add button to display the list of configured modules in the PC station. 3. Select the CP card that you want to configure as a submodule, and click OK. 4. Click OK on the WinLC Properties dialog and on the Edit Component dialog.
	<p>The CP card moves from the virtual rack of the PC station to the submodule slot of the WinLC controller. (This configuration may take several seconds.)</p> <p>For multiple CP cards, repeat the above steps as needed.</p>

You can select an occupied interface slot (IF slot) and click the Edit button to change the interface slot assignment for a configured CP card, or to change its name.

Notice

Any change to the configuration of a submodule causes WinLC to delete the control program. You must then download a new control program and configuration.


For example: Moving the submodule CP card to a different slot, WinLC recognizes that the hardware configuration has changed and deletes the control program, even if you had reconfigured the CP card for the same IF slot in the configuration of the PC station. You must still reload the control program from STEP 7.

Do not use the Set PG/PC Interface dialog to change the assignment of the CP card that has been configured as a submodule of WinLC.

Notice

Using the Set PG/PC Interface dialog to change the Interface Parameter Assignment of a CP card that has already been configured as a submodule causes an error that prohibits WinLC from going to RUN mode.

To recover from this error, perform the following steps:

1. Shut down WinLC.
2. Use the WinLC Properties dialog to delete the submodule from WinLC:
 - Double-click the computer icon  on the Windows taskbar to display the Station Configuration Editor
 - Select WinLC and click the Edit button to display the Edit Component dialog and click the Properties button to display the WinLC Properties dialog.
 - Select the submodule and click the Delete button.
4. Use the procedure above to reassign the CP card as a submodule.
5. Restart WinLC.

You must now download a new control program and configuration.

Testing the Submodule CP Card


Goal: To verify that the submodule CP card is configured correctly.

This test is especially important if you have installed more than one CP card in your computer.

Prerequisite: You must start the WinLC controller.

Software Tool: You use the Station Configuration Editor of SIMATIC NET.

After you start WinLC, you can check the operation of the submodule CP card:

1. Double-click the computer icon  on the Windows taskbar. The Station Configuration Editor appears.
2. Select the WinLC station and click the Edit button to display the Edit Component dialog.
3. Click the Properties button to display the WinLC Properties dialog.
4. Select the interface slot (IF slot) containing the CP card to be tested.
5. Click the Ring ON button.

The LEDs on the CP card at the back of your computer flash in an alternating pattern so you can verify that you have configured the correct CP card. The computer also emits an audible beep if the CP card is functioning.

6. Click the Ring OFF button to end the test of the CP card.

Viewing the Diagnostics for the CP Card

You can select an occupied interface slot (IF slot) and click the Diagnostics button to view communication information for a CP card. The Submodule Network Diagnostics dialog displays the current version of the selected CP card and the bus parameters.

You can also see a display of all of the nodes on the communication network, and the status of each one. You must click the Update button to build this display, because querying each node puts an additional load on the communication network.

Sub-Module Network Diagnostics - CP 5613

Status/Network Diagnostics

Interface: IF1
 CP5613_5614(PROFIBUS)<3>
 The device is operating in interrupt mode.

Bus Parameters	Value
Highest Station Address	126
Station address of this station	2
Baud rate in bits per second	12.0 Mbps
Target token rotation time	66035
Minimum Tsd	11
Maximum Tsd	800

Version:
 Siemens AG; CP 5613 EL (E2); FW: V 6.0.; HW: 5.0+

Bus Nodes

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
20	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
40	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
60	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
80	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
100	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
120	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Update

Key
 passive
 active
 active ready

OK

Configuring the Controller in STEP 7

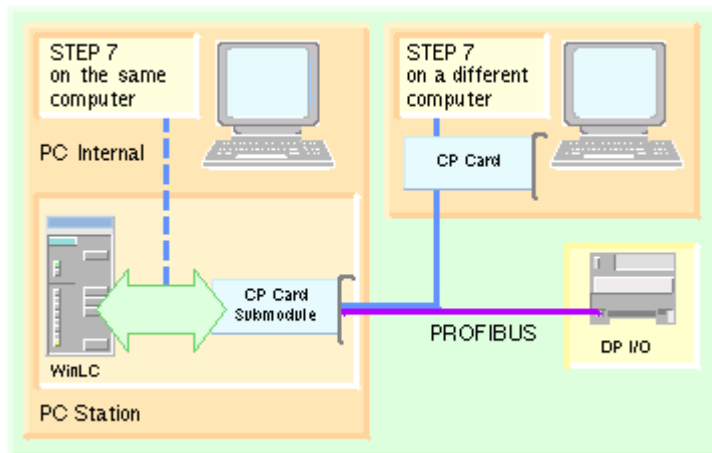
Use Case: Connecting STEP 7 to WinLC

WinLC uses two different network protocols:

- Communications with STEP 7, SIMATIC HMI products, and peer-to-peer communications with other controllers
- PROFIBUS-DP for communicating with the I/O devices on the DP subnet

In order for WinLC to function as an S7 controller, it must be able to communicate both with STEP 7 and with the distributed (DP) I/O.

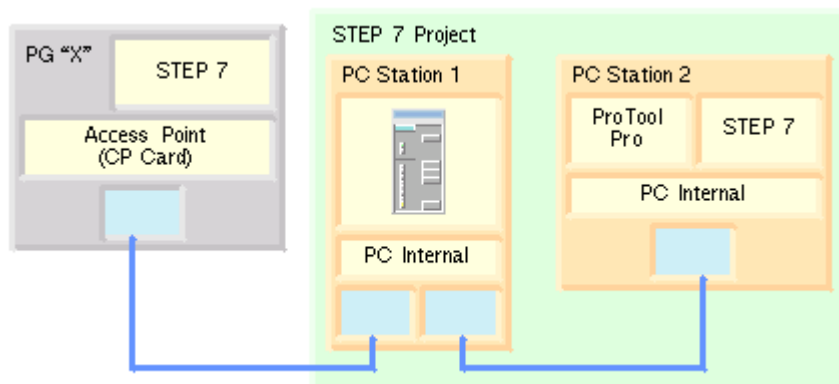
The following figure shows how WinLC communicates with distributed I/O over PROFIBUS-DP and with STEP 7 over MPI.



Configuring Communications between STEP 7 and WinLC

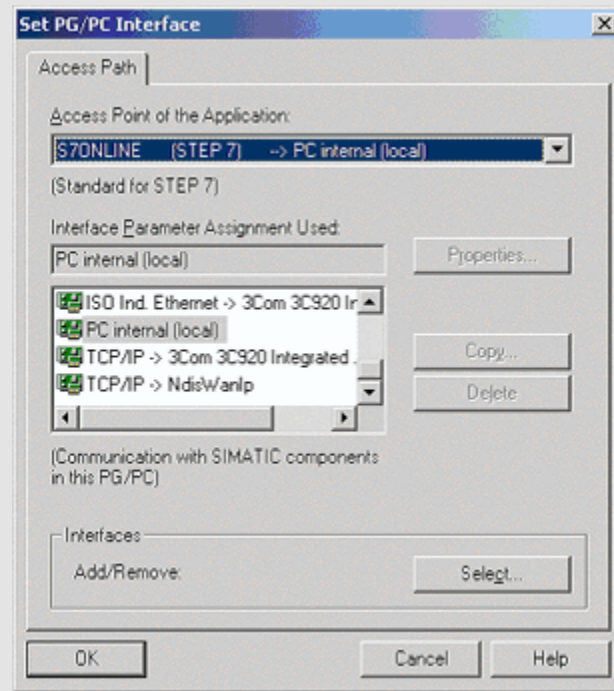
As shown in the following figure, you set the PG/PC interface to the appropriate access point to configure STEP 7 for communicating with WinLC:

- For a SIMATIC PC station within the STEP 7 project, select the PC Internal access point.
- For a programming device (PG or PC) somewhere on the network (with no Station Manager), select the access point for the CP card in the programming device.

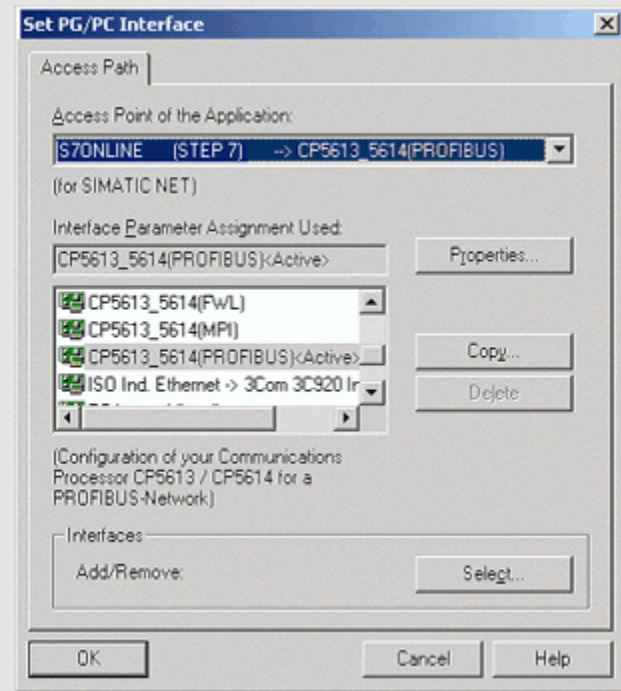


STEP 7 on the same computer or on a PC station within the same STEP 7 project

Set the S7ONLINE access point to PC Internal.

**STEP 7 on a different computer and not part of the STEP 7 project**

Set the S7ONLINE access point to the CP card that is used for S7 communications.



WinLC can also use the S7 communication protocols (MPI, TCP/IP, Industrial Ethernet, or PROFIBUS) for peer-to-peer communication with other controllers.

Configuring Communications between WinLC and the DP I/O

To configure WinLC for communicating with the DP I/O, you must designate one or more of the CP cards as submodules of WinLC. Configuring a CP card as a submodule of WinLC allows WinLC to use the CP card for communicating over both MPI and PROFIBUS-DP subnets.

Notice

Do not assign a submodule CP card for direct use by applications other than WinLC. For example: if you designated a CP card as a WinLC submodule, do not assign S7ONLINE to any parameter setting for that CP card.

Failure to observe this restriction may result in an inability to configure the CP card when WinLC is downloaded.

Use Case: Hardware Configuration in STEP 7

Goal: To configure the PC station in STEP 7 and to create the system blocks for WinLC.

Prerequisite: You must have installed both WinLC and STEP 7.

Software Tool: You use STEP 7 to create and download the configuration and the system blocks.

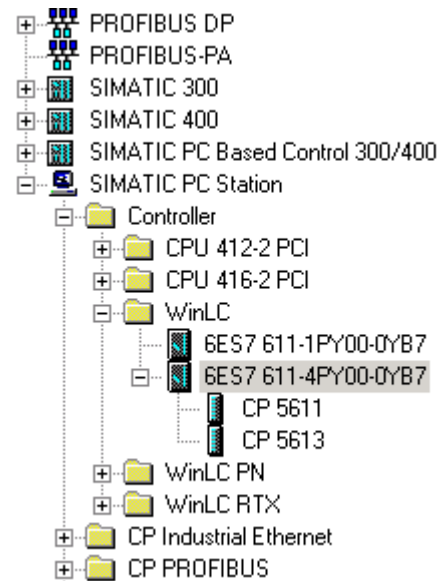
Next Step: You have finished with the configuration tasks and are ready to design and test your control program.

You configure the WinLC project in STEP 7 as you would for any S7 hardware controller. Refer to the STEP 7 help system and documentation for detailed information.

1 Using SIMATIC Manager

Create the project and PC station:

1. Select the **File > New** menu command to create a new project.
2. Select the **Insert > Station > SIMATIC PC Station** to insert a PC station into the project.
3. Change the name of the PC station to match the name of the target PC station defined in the Station Configuration Editor on the target computer. To find the station name, open the Station Configuration Editor and click the Station Name button.
4. Open the PC Station folder and double-click the Configuration icon to invoke the Hardware Configuration application.



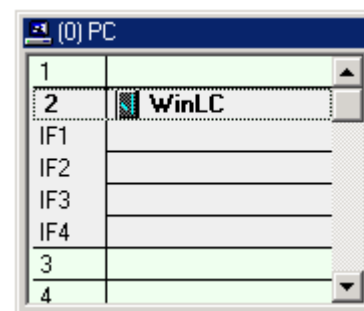
2 Using the Hardware Configuration application

Drag components from the Hardware Catalog to specify the hardware and software configuration for the PC station.

- Insert the WinLC controller (part number 6ES7 611-4PY00-0YB7) in the same index as for the target PC station that you configured with the Station Configuration Editor on the target computer.

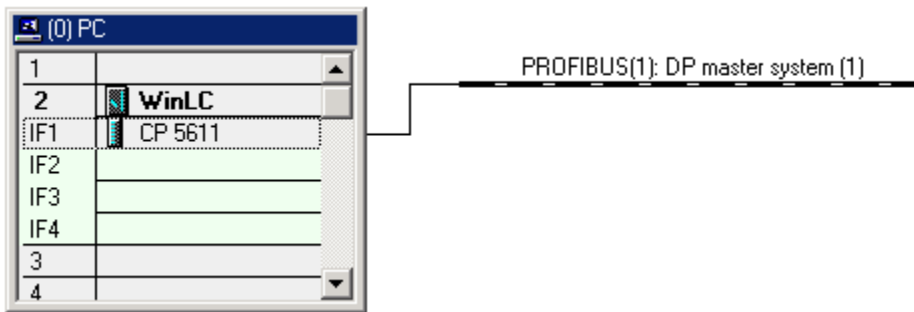
Verify that the name of the WinLC controller matches the name of the WinLC controller that you configured in the target PC station.

- Insert the submodule CP cards for the WinLC controller. The WinLC folder in the Hardware Catalog lists the CP card types that are available.



If you use a CP 5611 card as a submodule of WinLC, note that you can insert only one CP 5611 in the PC Station.

The submodule CP cards do not have to have the same name as in the PC Station configuration. However, this is recommended. They must have the same type and interface (IF) number as specified in the PC Station.



- Configure the distributed (DP) I/O for each of the submodule DP networks.
- Optional:** Insert any CP cards that are not used as submodules for the WinLC controller. These are the CP cards installed in the virtual rack of the PC station.
- Optional:** Insert any HMI devices.
- Optional:** Configure WinLC for peer-to-peer communications:
 1. Select the controller name in the SIMATIC Manager.
 2. Double-click the Connections icon in the right-hand pane.
 3. Use NET-PRO to describe the network.

After you have configured WinLC, you can use SIMATIC Manager to develop and to download your control program.



Caution

Downloading a control program that is too large for the memory of the computer can lock up the computer or cause the operation of WinLC to become unstable. Non-responsive or non-deterministic operations can cause damage to equipment and/or injury to personnel.

Although STEP 7 and WinLC do not limit the number of blocks or the size of the control program, your computer does have a limit, based on the available disk space and RAM memory. The limit for the size of the control program and number of blocks for your computer can only be determined by testing a configured system against the requirements of your control application.

After you have downloaded your program to WinLC, you can start the controller and use STEP 7 to monitor and modify the process variables.

Notice

Any change to the configuration of a submodule causes WinLC to delete the control program. You must then download a new control program and configuration.

For example: Moving the submodule CP card to a different slot, WinLC recognizes that the hardware configuration has changed and deletes the control program, even if you had reconfigured the CP card for the same IF slot in the configuration of the PC station. You must still reload the control program from STEP 7.

Invalid Characters

You can use STEP 7 to create a name for the WinLC controller and download the configuration with the new name to the controller in the Station Manager. However, you cannot use invalid characters for the name of the controller.

Caution

Using an invalid character in the WinLC name creates an instance of the controller that cannot be restarted.

Downloading a configuration that uses an invalid character in the WinLC name creates an invalid instance of the controller. This invalid instance will continue to run and will remain connected to STEP 7 until you shut down the controller. However, the desktop icon and the Start menu command will be removed. Without the desktop icon or Start menu command, you cannot restart the controller after it has been shut down.

Avoid the use of the invalid characters in controller names.

If you inadvertently downloaded a name that contains an invalid character:

1. Using STEP 7 HW-Config, rename the WinLC RTX controller to the previous valid name (the name prior to downloading the invalid name).
2. Download the configuration with the previous valid name to the PC station (even if the controller is not running).

After downloading the valid name for the controller, the desktop icon and the Start menu command reappear. You can now rename the controller to a new name that does not use invalid characters.

Character	Name															
/	Forward slash															
-	<p>Hyphen (also called a dash or a minus sign)</p> <p>You cannot create a name that begins with a hyphen (-). You can, however, use a hyphen within the name of the controller.</p> <table border="1"> <thead> <tr> <th>Name</th> <th>Valid?</th> <th>Explanation</th> </tr> </thead> <tbody> <tr> <td>Pump-1</td> <td>Valid</td> <td>Using a hyphen in the middle of the name is valid.</td> </tr> <tr> <td>Pump1-</td> <td>Valid</td> <td>Using a hyphen at the end of the name is valid.</td> </tr> <tr> <td>-Pump1</td> <td>Invalid</td> <td>Starting a name with a hyphen is not allowed.</td> </tr> <tr> <td>-</td> <td>Invalid</td> <td>Using a hyphen as a one-character name is not allowed.</td> </tr> </tbody> </table>	Name	Valid?	Explanation	Pump-1	Valid	Using a hyphen in the middle of the name is valid.	Pump1-	Valid	Using a hyphen at the end of the name is valid.	-Pump1	Invalid	Starting a name with a hyphen is not allowed.	-	Invalid	Using a hyphen as a one-character name is not allowed.
Name	Valid?	Explanation														
Pump-1	Valid	Using a hyphen in the middle of the name is valid.														
Pump1-	Valid	Using a hyphen at the end of the name is valid.														
-Pump1	Invalid	Starting a name with a hyphen is not allowed.														
-	Invalid	Using a hyphen as a one-character name is not allowed.														

Controller Operations

Starting and Shutting Down the Controller

The controller operates independently from the control panel:

- Closing the panel does **not** shut down WinLC.
- Shutting down WinLC does **not** close the panel.

The following settings affect the starting or shutting down of the controller:

- Selecting the Autostart option
- Configuring the controller for start at PC boot

Starting WinLC


If the control panel is not open, use one of the following methods to start the control panel and WinLC:

- Select the **Start > Simatic > PC Based Control** menu command. Then select the name of your WinLC controller. (After you have downloaded the control program to your WinLC, the name in the menu matches the name in STEP 7.)
- Double-click the desktop icon for WinLC: 

If the control panel is open, but WinLC is shut down, select the **CPU > Start Controller** menu command.

Shutting Down WinLC

Select the **CPU > Shut Down Controller** menu command to shut down the WinLC controller. This action does not close the control panel. This command is only available from the control panel when the controller is operating. After you shut down the controller, you can still change customization options.

An icon  is displayed in the Windows taskbar whenever the controller is operating. When the controller is operating and the control panel is closed, you can double-click this icon to open the control panel.

Closing the Control Panel

Select the **File > Exit** menu command to close the control panel.

Note: Closing the control panel does not shut down the controller.

Changing the Operating Mode of the Controller

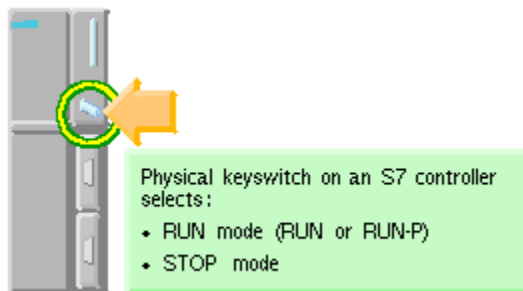
The control panel provides mode buttons that allow you to change the operating mode of WinLC between RUN mode and STOP mode. Like the other S7 controllers, WinLC also provides a RUN-P option that allows STEP 7 to interact with the controller in RUN mode. By clicking the appropriate mode button (or selecting the appropriate command from the **CPU** menu), you change the operating mode of the controller either to RUN mode (choosing either the RUN or RUN-P options) or to STOP mode.

The mode buttons correspond to the keyswitch positions of an S7 hardware controller:

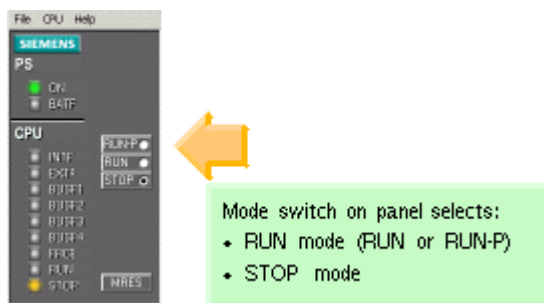
- **RUN-P:** The controller executes the control program. You can use STEP 7 to interact with the control program and to change the operating mode of the controller.
- **RUN:** The controller executes the control program. You can use STEP 7 to monitor the control program, but you cannot use STEP 7 to change the variables, reset the memory (MRES), or change the operating mode of the controller.
- **STOP:** The controller does not execute the control program. Outputs are set to their safe states. You can use STEP 7 to modify the control program and to reset the memory (MRES), but you cannot use STEP 7 to change the operating mode of the controller.

Relating the Mode Buttons and Status Indicators to the Operating Mode of the Controller

The mode buttons on the control panel function like the manual keyswitch on a hardware S7 controller.



For a hardware controller: The keyswitch does not change position if you use STEP 7 to change the operating mode, or if the controller automatically changes mode (for example, encounters an error condition that causes the controller to change from RUN mode to STOP mode).



For WinLC: The mode buttons (RUN-P, RUN, STOP) do not change if you use STEP 7 to change the operating mode, or if the controller automatically changes mode.

For both a hardware controller and WinLC, the RUN and STOP status indicators show the current operating mode of the controller. If the status indicator shows a different operating mode than the selected mode button, the controller has changed operating mode, possibly due to some error in the program or because you used STEP 7 to change the operating mode.

Using the Mode Buttons to Control Access to the Controller

You can use the mode button to allow or to prohibit access to the controller. As shown in the following table, the setting of the mode button limits the type of actions that can be performed.

Mode Button	Description
RUN-P	<p>Allowed:</p> <ul style="list-style-type: none"> • Uploading a program from the controller to your computer • Downloading a program to the controller • Downloading individual blocks to the controller • Using STEP 7 to modify program variables and to change the operating mode of the controller • Performing a memory reset from either WinLC or STEP 7 <p>The controller automatically goes to STOP mode when you reset the memory from WinLC. To perform a memory reset from STEP 7, you must first change the controller to STOP mode.</p> <p>Not Allowed:</p> <ul style="list-style-type: none"> • Archiving and restoring a control program
RUN	<p>Allowed:</p> <ul style="list-style-type: none"> • Uploading a program from the controller to your computer • Using STEP 7 programming software to monitor but not to modify variables • Using STEP 7 to change the operating mode • Performing a memory reset from WinLC <p>The controller automatically goes to STOP mode when you reset the memory from WinLC.</p> <p>Not Allowed:</p> <ul style="list-style-type: none"> • Downloading a control program or individual blocks to the controller • Using STEP 7 to modify variables in the program • Archiving and restoring a control program • Performing a memory reset from STEP 7
STOP	<p>Allowed:</p> <ul style="list-style-type: none"> • Uploading a program from the controller to your computer or programming device • Downloading a program or individual blocks to the controller • Performing a memory reset from either WinLC or STEP 7 • Archiving and restoring a control program <p>Not Allowed:</p> <ul style="list-style-type: none"> • Using STEP 7 to change the operating mode

Resetting the Memory Areas: MRES Command (CPU Menu)

The MRES (memory reset) command functions like a master reset of the controller by resetting the controller to its initial (default) state. The MRES command deletes the control program and the system data (configuration), and also disconnects any online communications (such as STEP 7, WinCC, PROFIBUS, or peer-to-peer).

Use one of the following methods to reset the memory:

- Click the MRES button on the control panel
- Select the **CPU > MRES** menu command
- Press the ALT+C+M keys

You can also use STEP 7 to perform a memory reset. However, you must use the mode buttons to change the controller to STOP or RUN-P mode before you can reset the memory from STEP 7.

The MRES command changes the controller to STOP mode, if necessary, and then performs the following tasks:

- Deletes the entire control program from both the work memory area and the load memory area. This includes OBs, DBs, FCs, FBs, and the system data.
- Resets the memory areas (I, Q, M, T, and C) to 0.
- Restores the default system configuration (for example, the size of the process-image areas, and the size of the diagnostic buffer).
- Deletes all active communications jobs (such as TIS) and all open communications.

The MRES command does not affect the submodule network addresses and does not affect the contents of the diagnostic buffer.

The STOP indicator flashes while the memory reset is in progress. After the memory has been reset, the diagnostics buffer is resized to its default size. Input (I) and output (Q) memory areas are also resized to their default sizes. After a memory reset, you may need to reconfigure these values to your own specifications.

You typically perform an MRES before downloading a new program to the controller. You **must** perform an MRES if the STOP indicator on the control panel is flashing slowly to alert you to one of the following conditions:

- Errors were detected in the work memory area, such as the size of the user program exceeds the work memory area
- A power cycle followed a defective state of the controller

Using the Status Indicators

You can use status indicators on the control panel to determine the current operating mode or to troubleshoot an error condition. These indicators correspond to the LED indicators on a hardware S7 PLC.

You cannot change the status of the controller by clicking the status indicators.

If the control program reaches a breakpoint set by the STEP 7 Program Editor, both the RUN and STOP indicators turn on while the breakpoint is active: the RUN indicator flashes, and the STOP indicator is on.

During a change from STOP mode to RUN mode, the RUN indicator flashes, and the STOP indicator is on. When the STOP indicator turns off, the outputs are enabled.

The table below describes the different status indicators for the control panel:

Indicator	Description
ON	Power supply. Lights up (solid) when you start the controller. Turns off when you shut the controller down.
BATF	Battery fault. Always off for the controller.
INTF	This indicator lights up (solid) to show error conditions within the controller, such as programming errors, arithmetic errors, timer errors, and counter errors. If the control program handles the error by executing OB80, OB121, or OB122, the INTF indicator goes off after 3 seconds if there is no subsequent error condition.
EXTF	This indicator lights up (solid) to show error conditions that exist outside of the controller, such as hardware faults, parameter assignment errors, communication errors, and I/O fault errors.
BUSF1 BUSF2 BUSF3 BUSF4	These indicators light up (flashing) to identify fault conditions in the communication with the distributed I/O. The number of the BUSF indicator corresponds to the IF number of the submodule that has a fault condition.
RUN STOP	Lights up (solid) to show the operating mode (RUN or STOP). When RUN is flashing and STOP is lighted (solid), the control program has reached a breakpoint. (RUN light blinks with 0.5 Hz.) Note: The RUN and STOP indicators show the actual operating mode of the controller. The RUN, RUN-P, and STOP mode buttons show the selected mode (similar to the keyswitch position on a hardware PLC), which can differ from the operating mode. For example: Changing the operating mode with STEP 7 causes the status indicators to change, but the mode buttons do not change.

Corrective Action If the STOP Indicator is Flashing Slowly

If the STOP indicator flashes slowly, the controller is requesting that you perform a memory reset (MRES). To recover from this condition, you must use the MRES command to reset the controller.

Corrective Action If All Status Indicators Are Flashing

If all of the status indicators are flashing at the same time, the controller is in a defective state and has encountered an error condition that cannot be fixed by resetting the memory with the MRES command. To recover from this condition, you must perform the following tasks:

1. Shut down the controller.
2. Restart the controller. The STOP indicator flashes with the RUN indicator off.
3. Use the MRES command to reset the memory.
4. Use STEP 7 to download the control program and system configuration, or to restore an archived control program.

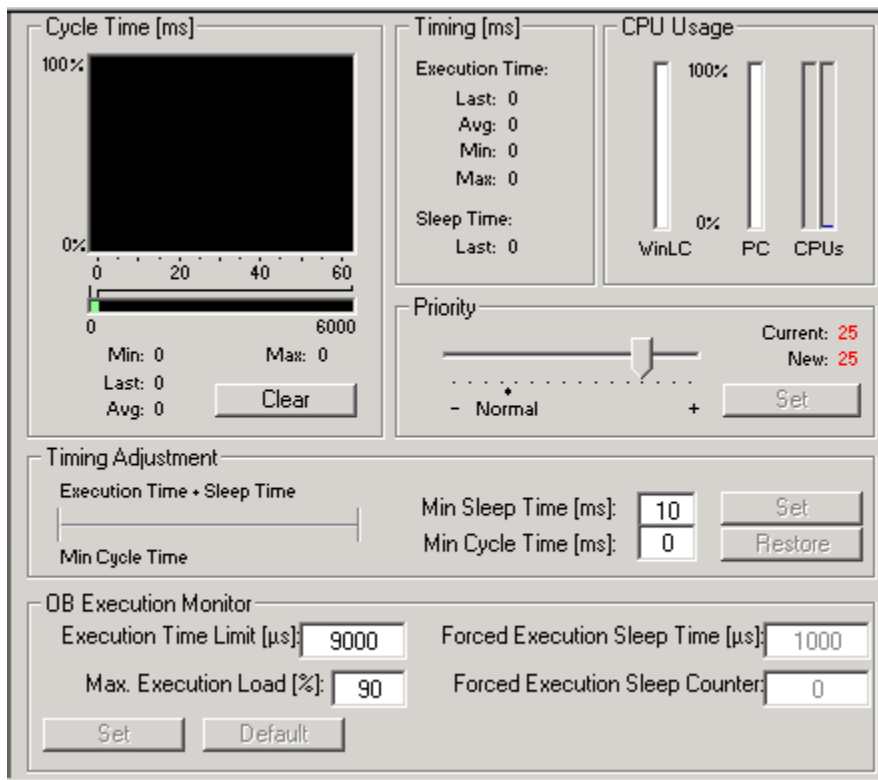
If either shutting down or restarting the controller does not resolve the problem, you may need to reboot your computer.

Using the Tuning Panel

You can use the tuning panel to view and adjust the current performance of the controller. The tuning panel displays information about the scan cycle, such as the execution time and the sleep time. By adjusting these values, you can tune the performance of the controller.

Note: The tuning panel is designed for adjusting the parameters and verifying the performance for WinLC. Because the tuning panel causes an additional load on the computer resources, do not leave the tuning panel open during normal operation of WinLC.

To open or close the tuning panel, select the **CPU > Tuning Panel** menu command. WinLC opens the tuning panel, as shown below. Click a region on the tuning panel picture for more information about its functionality.



Values other than the minimum cycle time are unique to WinLC and are not stored in the system configuration. Using the tuning panel to enter a value for minimum cycle time does not change the configuration of the controller.

Changing the controller from STOP mode to RUN mode resets the minimum scan cycle time parameter to the value that you configured in STEP 7. To make any changes made with the tuning panel permanent, you must use the STEP 7 Hardware Configuration tool.



Warning

Variation in the execution time or response time of the control program could potentially create a situation where the equipment or application being controlled can operate erratically and possibly cause damage to equipment or injury to personnel.

If the controller does not provide sufficient sleep time for the other applications to run, the computer can become unresponsive to operator input, or the controller and other applications can operate erratically. In addition, the execution of the control program can experience non-deterministic behavior (jitter) such that execution times can vary and start events can be delayed.

Always provide an external emergency stop circuit. In addition, always tune the sleep time and manage the performance of the controller so that your control program executes consistently.

The tuning panel contains the following functional areas:

Area	Description
Cycle Time	This area provides a histogram of execution times of the scan cycle over a 60-ms range. This histogram tracks minimum (shortest) and maximum (longest) scan times, as well as the percentage of scans that fall in various ranges of scan times. To delete the historical data and start a new histogram, click Clear. A STOP-to-RUN transition also resets the Cycle Time display, as does closing and reopening the tuning panel.
Timing	This read-only field displays the following information about the scan cycle: Execution Time displays the execution time for the last (most current) scan, the average scan cycle time, the shortest (minimum) scan cycle time, and the longest (maximum) scan cycle time. Sleep Time displays the amount of sleep time for the last (most current) scan.
CPU Usage	These read-only bar graphs show the percentage of the CPU used by WinLC and by the computer (PC). For a dual-processor system, there is a bar graph for each CPU.
Priority	Use this slider to set the priority level for the execution of WinLC relative to other applications running on your computer. Setting the priority higher means that the operating system responds to WinLC before executing lower-priority tasks. This results in less jitter in the start times and execution time of the OBs in your program.
Timing Adjustment	Use these fields to tune the scan cycle by entering values for the minimum sleep time and the minimum cycle time. These parameters determine the amount of sleep time that is added at the end of the free cycle. Click the Set button to apply these values. Click the Restore button to reset the values to those currently being used by the controller. After you apply new values, the panel stores these values for the controller and you can monitor the effect on the execution of your control program. To ensure that the minimum cycle time controls the sleep time for the controller, you must configure the scan cycle monitoring time and minimum scan cycle time parameters on the Cycle/Clock Memory tab of the Properties dialog box in STEP 7. Set the minimum scan cycle time to a value less than the value for scan cycle monitoring time. (The default scan cycle time is 6 seconds.)

Using the Diagnostic Buffer

Select the **CPU > Diagnostic Buffer** menu command to display the SIMATIC Diagnostic Buffer.

The Diagnostic Buffer allows you to view system diagnostic information without using the SIMATIC STEP 7 programming software and consists of an upper panel that displays an event list, a lower panel that displays specific event details, and controls for using the Diagnostic Buffer.

The diagnostic buffer is implemented as a ring buffer that contains single event entries. The events are displayed in descending order by time, with the most recent event at the top. If the ring buffer is full, a new event overwrites the oldest entry in the buffer.

The Diagnostic Buffer displays the following information:

No.	Time	Date	Event
1	10:10:22:295 am	08/29/02	New startup information in STOP mode
2	10:10:22:291 am	08/29/02	STOP caused by stop switch being activated
3	10:10:16:784 am	08/29/02	Mode transition from STARTUP to RUN
4	10:10:16:784 am	08/29/02	Request for manual warm restart
5	10:10:16:769 am	08/29/02	Mode transition from STOP to STARTUP
6	10:10:16:769 am	08/29/02	New startup information in STOP mode
7	10:10:12:741 am	08/29/02	New startup information in STOP mode
8	10:10:12:737 am	08/29/02	STOP caused by stop switch being activated
9	10:10:05:944 am	08/29/02	Mode transition from STARTUP to RUN
10	10:10:05:944 am	08/29/02	Request for manual warm restart
11	10:10:05:929 am	08/29/02	Mode transition from STOP to STARTUP

Details on Event: 3 of 120 Event ID: 0x4302

Mode transition from STARTUP to RUN
 Startup information:
 - Time for time stamp at the last backed up power on
 - Single processor operation
 Current/last startup type:
 - Warm restart triggered by switch setting; last power on backed up
 Permissibility of certain startup types:
 - Manual warm restart permitted

Format: Text Hex

Buttons: Update, Save, Help on Event, Help

Event List (upper panel): A list of all the events in the diagnostic buffer. The following information is shown for each diagnostic event:

- The number of the entry
- The date and time of the diagnostic event
- A brief description of the event

Event ID (between the upper and the lower panels): Displays the ID number of a selected event.

Event Details (lower panel): Displays the event details in either text or hexadecimal format.

If you have chosen Text format, the following details about a selected event appear in the lower panel:

- A brief description
- Additional information, depending on the event, such as the address of the instruction that caused the diagnostic event and the mode transition that was caused by the event
- The event state (incoming or outgoing)

If a single parameter of text cannot be identified, the diagnostic buffer displays the string "###". If no text exists for new modules or new events, the event numbers and the single parameters are displayed as hexadecimal values.

If you have chosen Hexadecimal format, the hexadecimal values of the selected event appear in the lower panel.

Sorting Events (upper panel)

You can sort the events listed in the upper panel by clicking the specific column:

- Number (determined by time and date)
- Event description

Choosing Format (lower panel)

You can display the diagnostic information in the lower panel in text or hexadecimal (Hex) format. In Hex format, the hexadecimal values of the 20 bytes of the selected event are displayed. To select the format:

- Click Text to display the event details in text format.
- Click Hex to display the hexadecimal values of the event.

Saving the Diagnostic Buffer

To save a text file containing the event list and the detailed information for every event, click the Save button. The text file contains the information either in text or in hexadecimal format.

Displaying Help

To display help on the diagnostic buffer, click the Help button. To display help on a specific event:

- Select the event in the upper panel.
- Click the Help on Event button.

Archiving and Restoring Control Programs

You can save the configuration and control program to an archive file (*.wld). You can use this archive file like the removable memory cartridge of a hardware PLC. The archive file allows you to easily restore the configuration and control program for the controller.

Note: A project or program from a release of WinLC V3 or earlier must be updated to a WinLC V4 project. Use STEP 7 to create a new project and to create a new configuration for the WinLC V4 controller and any submodules.

You can only archive or restore a control program when the controller is in STOP mode. You cannot archive or restore a control program when the controller is in RUN mode or is shut down.

Saving the configuration of the controller and the control program to an archive file makes it easy to restore the controller after a memory reset. However, the archive file does not function like the EEPROM cartridge of the hardware PLC in that the controller does not automatically restore the archive file after a memory reset (MRES). You must manually restore the archive file.

Creating an Archive File

An Archive file stores the current control program, the current system configuration, and the current values of the DBs. The Archive file does **not** store the configuration of the PC station.

To create an Archive file, select the **File > Archive** menu command. This command displays the Archive Active File dialog, which allows you to give a name to the file. The controller then creates the archive file with the extension .wld.

You can also use the SIMATIC Manager of STEP 7 to create an Archive file. Select the **File > Memory Card File > New** menu command.

Restoring an Archive File

When you restore an archive file, you reload the configuration and the control program for the controller. You can only restore archive files of extension .wld.

Before you can restore an archive file, you must set the controller to STOP mode. Use the following procedure to load an archived configuration and control program:

1. Click the STOP button to place the controller in STOP mode.
2. Select the **File > Restore** menu command.
3. Select the specific archive file to restore and click OK.

Storing Retentive Data

Storing Information About the Controller

WinLC stores the following operational information in the active file on your hard disk:

- Load memory stores the system data (configuration of the controller) and the initial values of the blocks of the control program.
- WinLC stores the state of the controller, which includes the last transition of the operating mode (STOP, RUN, or STARTUP) for the controller and the setting for the mode button (STOP, RUN, or RUN-P).
- As part of the shutdown process, WinLC creates the power-down state of the controller. The power-down state includes the contents of the diagnostic buffer, the current values for the retentive memory areas of the controller (such as timers, counters and bit memory), and the current values for the data blocks (Work memory).

WinLC updates these areas during operation and uses this information when starting up the controller.

Load Memory

When you download the control program from STEP 7, WinLC saves the blocks of the control program (program blocks and system data) in the Load memory area. These blocks include the initial values for the process variables used by the control program.

SFC82 (CREA_DBL) allows you to create new blocks in Load memory during the execution of the control program. You can use SFC84 (WRIT_DBL) to modify these blocks. The blocks created by these SFC82 are stored in the Load memory at the time that SFC82 runs.

Note: Data blocks (DBs) created by SFC22 (CREAT_DB) and SFC85 (CREA_DB) are not saved in the Load memory. These DBs are stored only in the Work memory.

Retentive Data

When you configure WinLC in STEP 7, you can determine the ranges of retentive data for the timers (T memory), counters (C memory), bit memory (M memory), and data blocks (DBs).

WinLC saves the retentive memory areas as part of the power-down state. If the power-down state was saved and the controller performs a warm restart (OB100), WinLC restores the retentive memory areas, including the values of the DBs stored in Work memory.

State of the Controller

WinLC stores the current operational status of the controller and updates the status on the following events:

- Whenever the controller changes operating mode (RUN to STOP, STOP to STARTUP, or STARTUP to RUN), WinLC updates the state of the controller to show the latest transition.
- Whenever the mode button on the control panel changes (STOP, RUN or RUN-P), WinLC updates the state of the mode button to show the latest action.

Power-Down State

Under normal circumstances, WinLC saves the current state of the controller and the retentive data, including the DBs from Work memory, when you shut down the controller. (The shutdown process can be initiated by the **CPU > Shut Down Controller** menu command or by shutting down the Windows operating system, either by user action or by UPS signal.) WinLC stores the following information from the Work memory:

- Values for the retentive data in the S7 memory areas (such as T, C, M, and DB)
- Diagnostic buffer

Note: The power-down state (which includes the diagnostic buffer) is **not** saved in cases when WinLC terminates abnormally, such as when the computer loses power (either by turning off the power or by power failure), or when WinLC is not able to write to the hard disk, such as following a Windows crash ("Blue Screen").

The power-down state is saved at the time WinLC shuts down. After WinLC starts up, the power-down state is loaded into WinLC and then is deleted (to avoid problems in case of an abnormal termination of the controller).

Shutting Down the Controller Saves the Retentive Data

The following table shows the actions that cause WinLC to save the retentive data.

Retentive Data	Action That Causes WinLC to Save This Data
Load memory (blocks and initial values of the control program, DBs, and system data)	Downloading the control program from STEP 7
State of the controller	Changing the operating mode (RUN to STOP, or STOP to RUN), either from STEP 7 or from the control panel Successfully shutting down WinLC
Power-down state	Successfully shutting down WinLC
Work memory	Successfully shutting down WinLC
Retentive memory areas (T, C, M, and DBs)	Successfully shutting down WinLC
Diagnostics buffer	Successfully shutting down WinLC

Restoring Memory Areas on Start Up

When starting the controller, WinLC searches the active file for the power-down state to determine whether the controller had been shut down correctly and performs the following tasks:

- WinLC restores the downloaded blocks of the control program from the Load memory.
- If WinLC finds the power-down state (showing that the controller had been shut down properly and successfully), WinLC updates the Work memory from the power-down state and loads the current values (at the time that the controller was shut down) for the retentive data.
- If WinLC does **not** find the power-down state (showing that the controller had **not** been shut down correctly), WinLC restores the Work memory to its initial state from Load memory (as downloaded from STEP 7).
- WinLC restores state of the controller, based on the saved operating mode and the Autostart configuration, and resets the mode button setting on the control panel.

If WinLC cannot read some element of the Load memory, state of the controller, or the power-down state), WinLC starts an unconfigured (empty) controller.

Restoring Memory Following a Successful Shutdown of the Controller

If the power-down status was successfully saved when shutting down the controller, WinLC restores the operational data for the controller:

- WinLC uses the data stored in the power-down state to restore the controller. This includes the retentive S7 memory areas, the current values of the process blocks (Work memory), and the contents of the diagnostic buffer.
 - Note:** If you configured the controller for a cold restart (OB102), WinLC resets the process variables and S7 memory areas to the initial values from the Load memory.
- Based on the Autostart settings, WinLC restores the state of the controller to either STOP mode or RUN mode.
 - Note:** If the controller was configured for Autostart, WinLC generates a startup event that identifies the type of startup: buffered or unbuffered. (An unbuffered startup is like reloading the control program from an EPROM file.) You can program OB100 to read this start event. For an unbuffered startup, the variable OB100_STOP at address LW6 is set to W#16#4309.
- WinLC restores the mode button to the setting when the state of the controller was last saved.

After restoring and starting the controller, WinLC deletes the Power-Down Status from the retentive memory area.

Restoring Memory if the Controller Was Not Shut Down Properly

If the controller was not shut down properly, WinLC does not create the power-down state.

Note: If the power-down state was not created, the diagnostic buffer is not saved. When you restart the controller, the diagnostic buffer will be empty.

If the power-down state was **not** saved when shutting down the controller, WinLC performs the following tasks when restarting the controller:

- WinLC reads the Load memory and restores the system configuration, the process variables and S7 memory areas to the initial values configured in STEP 7.
- WinLC reads the state of the controller and performs an unbuffered startup. (An unbuffered startup is like reloading the control program from an EPROM file. WinLC generates a startup event that you can program the OB100 to read.) Based on the Autostart settings, WinLC sets the controller to either STOP mode or RUN mode.
- WinLC restores the mode button to the setting when the controller was shut down.

Encountering Problems When Starting the Controller

If WinLC cannot read (or encounters an error) some element of the retentive memory area (Load memory, state of the controller, or power-down state), WinLC starts with an unconfigured (empty) controller. In this case, the controller is set to STOP mode with the mode button set to STOP, and the system data and control program are deleted.

Possible causes for this problem include a hardware error in your computer, or a partial block in the Load memory area caused by an error that occurred when WinLC was writing a block to the Load memory.

To recover from this condition, you must reload your control program and system data from STEP 7.

Note: The mode button of the controller is set to STOP mode. You can download the control program and system data from a remote computer, but you cannot use the remote computer to set the controller to RUN mode. You must go to the local computer for WinLC and set the mode button to RUN or RUN-P in order to place the controller in RUN mode.

Using SFCs to Retain Data

You can use SFC82 (CREA_DBL), SFC83 (READ_DBL), and SFC84 (WRIT_DBL) to save data at significant events in your process. For example, you may want to store the recipe values into Load memory when changing a recipe without downloading new blocks for the control program.

SFC82 and SFC84 modify the data for the control program that is stored in the Load memory. Saving the blocks to Load memory (instead of keeping the values in Work memory) ensures that these blocks are available even if WinLC cannot save the power-down state when shutting down the controller.

SFC82 (CREA_DBL), SFC83 (READ_DBL), and SFC84 (WRIT_DBL) create and update blocks that are stored as part of your control program in Load memory. The blocks are created or modified in Load memory when your control program executes these SFCs.

SFC82, SFC83, and SFC84 are asynchronous SFCs that run in the background.

Note: If you call SFC82, SFC83, or SFC84 from the startup OB (OB100 or OB102), WinLC executes these SFCs synchronously. This differs from the operation of a hardware PLC.

Like the other asynchronous SFCs, SFC82, SFC83, and SFC84 are typically long-running SFCs that can require a relatively long time to complete. (The time for the SFC call itself will be short, but the actual operation for the SFC will be executing in the background.) In order to use asynchronous SFCs, you must allow sufficient sleep time to allow WinLC to process the SFCs without encountering jitter.

Note: Do not use a polling loop that looks for the completion of an asynchronous SFC, especially for SFC82, SFC83, or SFC84. Because the asynchronous SFC is being executed in the background, having your control program loop until the SFC finishes will extend the execution of the OB that is performing the polling loop and can cause jitter.

Caution

Whenever your control program calls SFC82, SFC83, or SFC84, the SFC reads or writes data to the disk. If you call these SFCs every scan (such as from OB1) or from a cyclical OB that is executing rapidly, the constant reading or writing to the disk can cause the disk to fail or can add jitter.

You should only call SFC82, SFC83, or SFC84 to record a significant process event, such as a change of recipe.

Uninterruptible Power Supply (UPS)

You can use a UPS system to provide emergency power for your computer. The UPS system can help ensure that WinLC shuts down correctly and saves the power-down state in case of a power failure.

Refer to the manufacturer's documentation for your computer and your UPS system.

Microsoft Windows provides a dialog for configuring the UPS for your computer:

1. Select the **Start > Settings > Control Panel** menu command to display the control panel.
2. Double-click the Power Options icon to display the Power Options Properties dialog.
3. Click the UPS tab and configure the parameters for your UPS system.
4. Click Apply or OK to set the UPS properties.

Customizing and Security Options

Customizing Options

To open the Customize dialog box, select the **CPU > Options > Customize** menu command. The tabs of the dialog box allow you to customize the control panel as follows:

General

Select **Always On Top** to display the control panel on top of all other open windows.

Language

The language field displays the current display language for the control panel.

The language select list displays the installed languages for the control panel. Click a language selection to change the control panel display language.

Note: To install the languages that are available for the control panel, run the setup program and select the languages from the dialog.

AutoStart

Select **Autostart CPU** to set the autostart feature. The autostart feature allows the controller to start automatically in RUN mode under the conditions described in *Selecting the Autostart Feature*.

Selecting the Language

You can change the display language for the control panel menus and online help.

To change the display language, follow these steps:

1. Select the **CPU > Options > Customize** menu command to display the Customize dialog.
2. In the Customize dialog, select the Language tab.
3. Select the language for the control panel.
4. Click Apply to change the language.
5. Click OK to close the Customize dialog.

The control panel automatically changes to the selected language.

Selecting the Autostart Feature

The panel provides an Autostart feature that restarts the controller in the same operating mode as when previously shut down. With the Autostart feature enabled:

- If the controller was in RUN mode when shut down, the controller restarts in RUN mode.
- If the controller was in STOP mode when shut down, the controller restarts in STOP mode.

If the Autostart feature is **not** enabled, the controller always starts in STOP mode.

Use the following procedure to enable the Autostart feature:

1. Select the **CPU > Options > Customize** menu command to display the Customize dialog.
2. In the Customize dialog, select the Autostart tab.
3. Select the Autostart CPU option for the Startup Mode.
4. Click Apply to enable the Autostart feature and click OK to close the Customize dialog.

Setting the Security Options

Select the **CPU > Options > Security** menu command to change security options. The control panel displays the Access Verification dialog. You must enter your password in this dialog in order to make any changes to the security settings for the controller.

Note: The default password is an empty field containing no characters. To enter the default password, press the Enter key.

Security Level

The Security dialog allows you to set levels of password security that limit access to the controller. The following security access options are provided:

- **Password:** When you select Password, certain control panel operations, such as changing the operating mode, archiving and restoring a control program, and opening the tuning panel, require that the user enter a password.
- **Confirmation:** When you select Confirmation, operating mode changes require that the user acknowledge a confirmation dialog box.
- **None:** When you select None, no confirmation or password is required.

Password Prompt Interval

You can set the Password Prompt Interval to a time interval of your choice, from 0 to a maximum of 23 hours, 59 minutes. After you have entered your password, you are not prompted for it again until this time interval has expired. The default setting of 0 means that you must enter a password for each protected operation.

Shutting down and starting the controller does not affect the expiration of the Password Prompt Interval; however, it is reset whenever you shut down the control panel. The next time you start the control panel and access a password-protected function, you will be prompted for password entry.

Change Password

Click the Change Password button to display the Change Password dialog.

Note: If you create a password, but set the security level to None (disabling the password), you still need to enter the configured password in the Access Verification dialog before you can access the Security dialog box again.



Warning

Running the controller without confirmation or password protection increases the risk that an operator may change the controller mode inadvertently, which could cause the process or equipment to operate unpredictably, resulting in potential damage to equipment and/or death or serious injury to personnel.

Exercise caution to ensure that you do not inadvertently change the operating mode, or permit unauthorized persons to access the machine or process. Always install a physical emergency stop circuit for your machine or process.

Changing the Password

The Change Password dialog allows you to change the current password.

Note: The default password is an empty field containing no characters. To enter the default password, press the Enter key.

Use the following procedure to change the password:

1. In the Old Password field, enter the old password.
2. In the New Password field, enter the new password (maximum length 12 characters).
3. In the Confirm New Password field, enter the new password again.
4. Click OK to apply all the changes made in this dialog, or click Cancel to reject all changes.

To subsequently access the security options, you must enter the password at the Access Verification dialog.

Startup Options for the Controller

Starting the Controller at PC Boot

By default, you must start the controller manually after the computer reboots. You can, however, register the controller to start automatically after a reboot.

Note: To configure the controller for starting in the same operating mode (STOP or RUN) as when previously shut down, use the Autostart feature.

Registering the Controller for Start at PC Boot

To register the controller to start automatically, follow these steps:

1. Shut down the controller.
2. Select the **CPU > Register Controller for Start at PC Boot** menu command.

WinLC will now start automatically whenever you start your computer.

Unregistering the Controller for Start at PC Boot

To unregister the controller for starting automatically, follow these steps:

1. Shut down the controller.
2. Select the **CPU > Unregister Controller for Start at PC Boot** menu command.

WinLC will now **not** start automatically whenever you start your computer. To start WinLC, you must manually start the controller.

Setting the Restart Method

The restart method determines which startup OB is executed whenever the controller changes from STOP mode to RUN mode. The startup OB allows you to initialize your control program and variables. WinLC supports two restart methods:

- **Warm restart:** The controller executes OB100 before starting the free cycle (OB1). A warm restart resets the peripheral inputs (PI) and changes the peripheral outputs (PQ) to a pre-defined safe state (default is 0). The warm restart also saves the current value for the retentive memory areas for the memory bits (M), timers (T), counters (C), and data blocks (DBs).
- **Cold restart:** The controller executes OB102 before starting the free cycle (OB1). Like a warm restart, a cold restart resets the peripheral inputs (PI) and changes the peripheral outputs (PQ) to a pre-defined safe state (default is 0). However, a cold restart does **not** save the retentive memory (M, T, C, or DB), but sets these areas to their default (initial) values.

You use STEP 7 to configure the default restart method for the controller. The default restart method is stored in the configuration (system data) for the controller that you download with your control program. WinLC uses this restart method when WinLC is configured for Autostart and returns to RUN mode following a power cycle.

Whenever you click (using the left mouse button) the RUN or RUN-P mode button on the panel to change from STOP mode to RUN mode, WinLC performs a warm restart, executing OB100.

To select a specific restart method, choose one of the following options for changing WinLC from STOP mode to RUN mode:

- Select either the **CPU > RUN** or **CPU > RUN-P** menu command to change the controller from STOP to RUN mode.
- Right-click (using the right mouse button) the RUN or RUN-P mode button.

Both of these actions display the Restart Method dialog that allows you to select either a warm or cold restart.

Note: If you have configured the confirmation security option, you must acknowledge a confirmation dialog before the control panel displays the Restart Method dialog.

If you have configured the password security option and the password prompt interval is either 0 or has expired, the control panel displays the Access Verification dialog for you to enter the password. After verifying successful password entry, the control panel displays the Restart Method dialog.

After executing OB100 (warm restart) or OB102 (cold restart) according to your selection, WinLC executes the free cycle (OB1).

STEP 7 Operations

Using STEP 7 with the Controller

STEP 7 provides programming and configuration tools for working with WinLC. You define the WinLC configuration through the Hardware Configuration tool of STEP 7, as well as in the Station Configuration Editor. You can develop a control program using any of the STEP 7 control programming languages, and download it to WinLC once you have established a connection. You also use STEP 7 to configure operational parameters and I/O addresses for the controller.

Accessing WinLC from STEP 7

To access WinLC from the STEP 7 programming software, follow these steps:

1. Using the SIMATIC Manager, open the project for WinLC.
2. Select the **View > Online** menu command to change to the "Standard Hierarchy, Online" view.

STEP 7 establishes an online connection to WinLC.

Configuring the Operational Parameters

Configuring the Operational Parameters for WinLC

STEP 7 provides a Hardware Configuration tool for configuring the operational characteristics for WinLC. This configuration is then stored in various SDBs in the System Data container.

After you download the System Data, WinLC uses the configured parameters for the following events:

- Whenever you start the controller
- On the transition to RUN mode (if you modified the hardware configuration online while WinLC was in STOP mode)

You use the Properties dialog to configure the operational parameters.

Properties - WinLC - (R0/S2)

Interrupts | Time-of-Day Interrupts | Cyclic Interrupts | Diagnostics/Clock

General | Startup | Cycle/Clock Memory | Retentive Memory | Memory

Short Description: WinLC

WinLC V4.1; Windows Logic Controller for Windows 2000/XP; DP connector (DP master); DPV1; routing; S7 Communication; PROFINet; firmware V4.1

Order No./ firmware: 6ES7 611-4PY00-0YB7 / V4.1

Name: WinLC

Plant designation:

Comment:

OK Cancel Help

Parameter	Description
General	Provides information about WinLC
Startup	Defines the operational characteristics of WinLC for powering on or going to RUN mode
Cycle/Clock Memory	Cycle: defines any constraints on the scan cycle (such as the minimum and maximum scan cycle time, the size of the process image, and program requirements relating to I/O access error reporting.) Clock Memory: defines a memory byte to function as a "clock memory"—each bit of this byte toggles on and off at a different frequency
Time-of-Day Interrupt	Configures the operation of the time-of-day interrupts (OB10)
Interrupts	Defines the priority class for the hardware interrupts (OB40), the time-delay interrupts (OB20), the DPV1 interrupts (OB55 to OB57), and the asynchronous error interrupts (OB82, OB83, OB85, and OB86)
Retentive Memory	Defines the memory areas (M, T, and C) to be retained following a power failure or a transition from STOP mode to RUN mode
Cyclic Interrupt	Defines the operation of the cyclic interrupts (OB32 to OB36)
Diagnostics/Clock	Defines the reporting of diagnostic errors and the synchronization and the correction factor for the WinLC clock
Memory	Defines the amount of local data (L memory) for each priority class and the maximum number of communication jobs that can be outstanding at one time.

Accessing Operational Parameters

To configure any of these operational parameters in STEP 7, open the SIMATIC Manager and follow these steps:

1. In the SIMATIC Manager, select the PC station.
2. Click the Configuration icon.
3. Right-click the WinLC icon in the Hardware Configuration rack and select Object Properties.
4. Click the tab with the name of the parameter that you want to configure (such as Cyclic Interrupt) and enter the appropriate values in the dialog.
5. Click OK to confirm your configuration.

Configuring the Startup Characteristics

You can use the Startup tab of the STEP 7 Hardware Configuration tool to configure WinLC to perform certain tasks before going to RUN mode. The following table lists the parameters for configuring the startup characteristics.

Parameter	Description	Range	Default
Startup when expected / actual configuration differ	Startup is allowed even if I/O is not configured properly	Yes/No	Yes
Startup after Power On	Specifies what type of restart to perform if you have selected the Autostart feature	Warm restart Cold restart	Warm restart
Finished message by modules	Specifies how long WinLC waits for configured S7 I/O modules to respond with a ready message	1 to 650 (100 ms intervals)	650 (65 s)
Transfer of parameters to modules	Specifies how long WinLC waits for transfer of parameters to S7 I/O modules following a ready message	1 to 650 (100 ms intervals)	100 (10 s)

You can also change the restart method from the WinLC panel for a particular change to RUN mode.

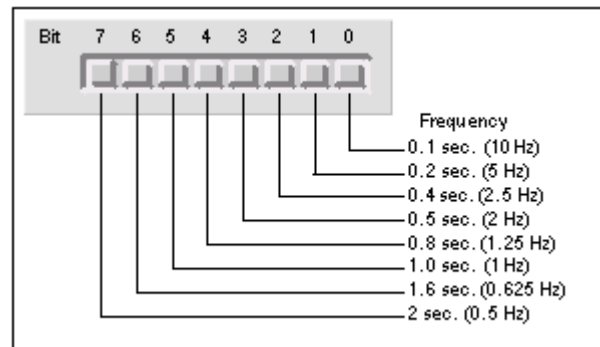
Configuring the Clock Memory

You can use the Cycle/Clock Memory tab of the STEP 7 Hardware Configuration to configure a byte of the bit memory (M) area to function as a "clock memory." The following table lists the parameters and ranges for configuring a clock memory.

Parameter	Description	Range	Default
Clock Memory	Enables the clock memory (if enabled, you must enter a memory byte address)	Yes or No	No
Memory Byte	Defines a memory byte (MB) to function as a clock memory	0 to MB2047	Disabled

When this byte has been configured as clock memory, the bits turn on and off (with a duty cycle of 1:1) at fixed frequencies. (The eight bits in the byte yield eight different, fixed frequencies.)

The figure shows the frequencies of the different bits for the byte used as clock memory.



Configuring the Scan Cycle

You can use the Cycle/Clock Memory tab in the Hardware Configuration tool of STEP 7 to configure certain aspects of the scan cycle. The table below lists the parameters for configuring the scan cycle.

Note: The minimum cycle time of WinLC includes both the time required for executing the control program and the sleep time (which allows your computer to perform other tasks).

WinLC monitors the execution time of the scan cycle. If the scan cycle (execution time of the control program plus the sleep time) exceeds the scan cycle monitoring time (watchdog), WinLC starts an error OB. The scan cycle monitoring time must be greater than the maximum execution time for the scan cycle plus the configured sleep time.

Parameter	Description
Update OB1 process image cyclically	<p>Indicates whether to update the process images as part of execution of OB1.</p> <p>Range: Selected / Unselected Default: Selected</p>
Scan cycle monitoring time	<p>Defines the maximum time for the scan cycle. (The value for the scan cycle includes both the execution time of the control program and the sleep time of the controller.)</p> <p>This value must be larger than the value for the minimum scan cycle time.</p> <p>The following list gives a few examples of events that could cause the controller to exceed the limit on maximum cycle time:</p> <ul style="list-style-type: none"> • Starting other PC applications • Processing an increasing number of interrupts in the program • PC blocked by high-priority drivers or by continuous hardware activities • Processing an error in the control program <p>Range: 1 to 6000 ms Default: 6000 ms</p>
Minimum scan cycle time	<p>Defines the minimum time for the scan cycle. (This value includes both the execution time of the control program and the sleep time of the controller.)</p> <p>The minimum scan time allows you to determine the percentage of processing time of your computer to dedicate to the controller. For example: if you entered a minimum scan time that is twice as long as the actual execution time of the user program, 50% of the processing time would be dedicated to WinLC and 50% could be used by another application (based on process priority).</p> <p>Range: 0 to 5999 ms Default: 0 ms</p>
Size of process image input area	<p>Defines the overall size of the process image input area</p> <p>Range: 0 to 8192 bytes Default: 512 bytes</p>
Size of process image output area	<p>Defines the overall size of the process image output area</p> <p>Range: 0 to 8192 bytes Default: 512 bytes</p>
OB85 Call Up	<p>Specifies handling of I/O access errors during process image reading or writing. The choices are:</p> <ul style="list-style-type: none"> • Do not call OB85 (Never) • Call OB85 when module first fails and again when module returns to normal operation (Coming/Going) • Call OB85 on every access while the module is failed (Always) <p>Range: Never, Coming/Going, Always Default: Always</p>

Configuring Retentive Memory Areas

You can use the Retentive Memory tab of the STEP 7 Hardware Configuration tool to configure the following areas of memory to be retained in the event of loss of power or on a transition from STOP mode to RUN mode:

- Number of Memory bytes: up to 2048 bytes (from MB0 to MB2047)
- Number of S7 timers: up to 512 timers (from T 0 to T 511)
- Number of S7 counters: up to 512 counters (from C 0 to C 511)

In the event of an abnormal shutdown while the controller is running, the current values for these areas, as well as the DBs, are lost. Otherwise, the values are retained according to the configured parameters shown in the table below.

Note: DBs that were created by SFC22 (CREATE_DB) or SFC85 (CREA_DB) are not retained following a cold restart. All other DBs are retentive following a normal shutdown.

The following table lists the parameters for configuring the retentive memory areas:

Parameter	Description	Range	Default
Memory Bytes	Enters the number of memory bytes to be retained (starting from MB0)	0 to 2048	16
S7 Timers	Enters the number of S7 timers to be retained (starting from T 0)	0 to 512	0
S7 Counters	Enters the number of S7 counters to be retained (starting from C 0)	0 to 512	8

Configuring the Time-of-Day Interrupt

WinLC supports one time-of-day interrupt (OB10). To configure OB10, use the Time-of-Day tab in the Hardware Configuration tool of STEP 7. The following table lists the parameters for the time-of-day interrupt.

Parameter	Description	Range	Default
Active	Determines whether OB10 is automatically activated following a warm restart	Yes/No	No
Execution	Selects the frequency for executing OB10	None Once Once per minute Hourly Daily Weekly Monthly End of the month Yearly	None
Start Date/Time	Determines the starting date and time for executing OB10 Date: day.month.year Time: hours:minutes:seconds (24-hour format)	Any valid time and date	01.01.94 00:00:00

Configuring the Interrupts

Use the Interrupts tab in the Hardware Configuration tool of STEP 7 to configure the priority class for the interrupt OBs supported by WinLC. You deselect an OB by configuring it with a priority class of 0. The following table lists the parameters for the different interrupts:

Parameter	Description	Range	Default
Hardware Interrupts: OB40	Defines the priority class and process image segment for the hardware interrupt OB	Priority class: 2 to 24 PI partition: None, 1 to 15	Priority Class: 16 PI partition: None
Time-Delay Interrupt: OB20	Defines the priority class and process image segment for the time-delay interrupt OB	Priority class: 2 to 24 PI partition: None, 1 to 15	Priority Class: 3 PI partition: None
DPV1 Interrupts: OB55, OB56, OB57	Defines the priority class for the DPV1 interrupt OBs	Priority class: 2 to 24	Priority Class: 2
Asynchronous Error Interrupts: OB82, OB83, OB85, and OB86	Defines the priority class for the asynchronous error interrupts	Priority class: 24 to 26	Priority Class: 25

Configuring a Cyclic Interrupt

WinLC supports cyclic interrupts OB32 to OB36. You can use the Cyclic Interrupts tab in the Hardware Configuration tool of STEP 7 to configure a time interval for executing a cyclic interrupt OB. The following table lists the parameters for the cyclic interrupt. To deselect a cyclic OB, configure the priority class to 0.

Parameter	Description	Range	Default
Priority	Determines the priority class of the cyclic interrupt	Priority class: 2 to 24	OB32: 9 OB33: 10 OB34: 11 OB35: 12 OB36: 13
Execution	Determines the time interval (in milliseconds) for executing the cyclic interrupt	0 to 60000	OB32: 1000 OB33: 500 OB34: 200 OB35: 100 OB36: 50
Phase Offset	Defines an amount of time (in milliseconds) that the start of a cyclic interrupt can be delayed in order to allow another cyclic interrupt to finish	0 to 59999	0
Process Image	Defines the process image segment to be updated automatically when the OB runs.	None, or 1 to 15	None

Based on the time interval that you configure, WinLC starts the execution of the cyclic interrupt OB at the appropriate time. The optimum time interval for your application depends on the processing speed of your computer and the execution time of the cyclic OB. Jitter can cause an occasional overrun in the start event for a cyclic OB, which might cause WinLC to go to STOP mode. Other factors that affect the execution of the OB include the following situations:

- The program in the OB takes longer to execute than the interval allows. If the execution of the program consistently overruns the start event of the cyclic OB, WinLC can go to STOP mode (unless OB80 is loaded).
- Programs in other priority classes frequently interrupt or take longer to execute, which causes the controller to not execute the cyclic OB at the scheduled time. If this occasionally causes an overrun, WinLC starts the cyclic OB as soon as the first OB finishes.
- STEP 7 performs some task or function that causes the controller not to execute the cyclic OB at the scheduled time.

The sleep time of the WinLC scan cycle does not affect the execution of a cyclic interrupt OB: WinLC attempts to execute the OB at the appropriate interval regardless of the amount of sleep time that you configure for the scan. WinLC provides several types of free cycle sleep management for managing sleep time. If a cyclic interrupt OB runs too frequently or requires too much of the time allotted for the total scan, it could cause the watchdog timer to time out (calling OB80 or going to STOP mode).

Configuring the I/O Addresses

Assigning Addresses for the DP I/O

You use the Hardware Configuration tool of the STEP 7 programming software to specify elements and addresses for the PROFIBUS-DP configuration. You configure the following addresses for each DP slave device on the network:

- Node address and diagnostic address for each node of the network
- Logical address range for the data of the I/O modules on each node

You must download the PROFIBUS-DP configuration to WinLC before you attempt to operate the PROFIBUS network.

Each node of the DP network has a unique node address. The DP master uses this node address to communicate with its DP slaves. The control program generally does not use the node address to reference a data for the slave device. Your control program uses the logical address of the I/O and the diagnostic address for the DP slave that you configured in STEP 7.

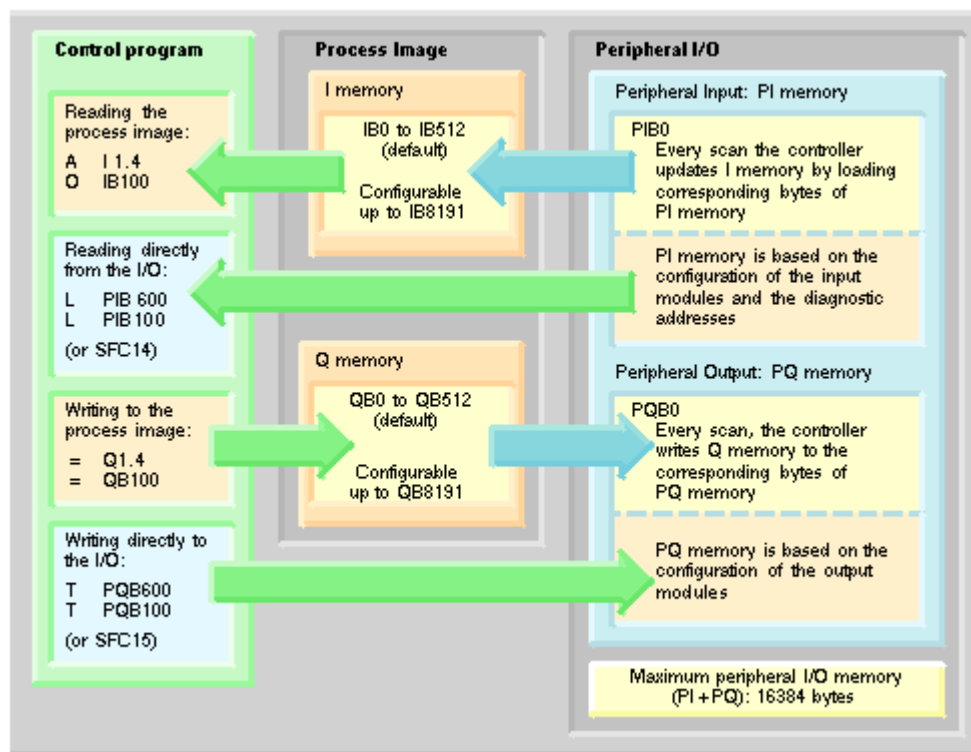
Like the other S7 controllers, WinLC provides memory areas for accessing the inputs and outputs

Peripheral Input Area (PI) and Peripheral Output Area (PQ)

When you configure the DP I/O in STEP 7, you assign a range of the PI or PQ memory for each I/O module. You also assign a diagnostic address for the I/O modules in the PI memory. The amount of the PI and PQ memory is determined by your DP I/O configuration, but the maximum amount of peripheral I/O memory (PI + PQ) is 16 Kbytes (16384 bytes).

Process-image Input Area (I) and Process-image Output Area (Q)

When you configure the operational parameters for WinLC in STEP 7, you determine the size of the I and Q memory areas. You can configure a maximum of 8 Kbytes (8192 bytes) for both I and Q memory (up to 8 Kbytes for I memory, and up to 8 Kbytes for Q memory). As part of the scan cycle, WinLC reads the PI memory into the corresponding I memory locations and writes the Q memory to the corresponding PQ memory locations.



Your control program can access the process image (reading from I memory and writing to Q memory). These values are updated every scan.

You can also access the peripheral I/O directly (reading from PI memory and writing to PQ memory) by using the Load (L) and Transfer (T) commands or by using SFC14 (DPRD_DAT) and SFC15 (DPWR_DAT).

The following table provides an overview of the addresses that can be assigned to the DP I/O.

Address Areas	Size
Process-image areas I and Q	<p>Inputs: 0 to 8192 bytes (configurable) Addressed as IB0 through IBn, where n is the configured input process image size minus 1.</p> <p>Outputs: 0 to 8192 bytes (configurable) Addressed as QB0 through QBm, where m is the configured output process image size minus 1</p>
Total amount for the distributed I/O (accessed by Load and Transfer instructions)	<p>16384 bytes</p> <p>This memory can be any mixture of peripheral inputs (PI) and peripheral outputs (PQ), based on the configuration of the distributed I/O (for both digital and analog I/O). However, the PI and PQ areas together cannot exceed the 16384-byte maximum size.</p> <p>Note: The diagnostic address for a slave device is configured in the PI memory.</p> <p>Addressed as:</p> <p>Inputs: PIB0 to PIBn, where n is the configured input size minus 1.</p> <p>Outputs: PQB0 to PQBm, where m is the configured output size minus 1</p>
Accessing consistent data	<p>Load (L) and Transfer (T) instructions access up to 4 bytes of consistent data</p> <p>SFC14 and SFC15 access up to 244 bytes of consistent data</p>
Maximum Input/Output data for one node	Up to 244 bytes

Accessing the Data in the DP I/O Modules

During the configuration of WinLC, you allocate a logical address in the Input (I) or Output (Q) area for the data of each I/O module of the DP network. You use these addresses to access the data for the I/O module. WinLC also uses the lowest logical base address for a module for reporting the module events to the control program.

The following table lists the methods for accessing the distributed I/O:

- To access data as bytes, words, or double words (that is, as 1 byte, 2 bytes, or 4 bytes), use the Load (L) and Transfer (T) instructions (for statement list) or the Assign Value (MOVE) instruction (for ladder logic) to read and write the distributed inputs and outputs. You can access the I/O data either from the process image (I) area or from the peripheral image (PI) area.
- To access data that has consistency of 3 bytes or more than 4 bytes (up to 244 bytes), use SFC14 (DPRD_DAT) and SFC15 (DPWR_DAT). SFC14 and SFC15 always access the peripheral image of the I/O module. Both SFC14 and SFC15 also allow you to access 1 byte, 2 bytes, and 4 bytes of data.

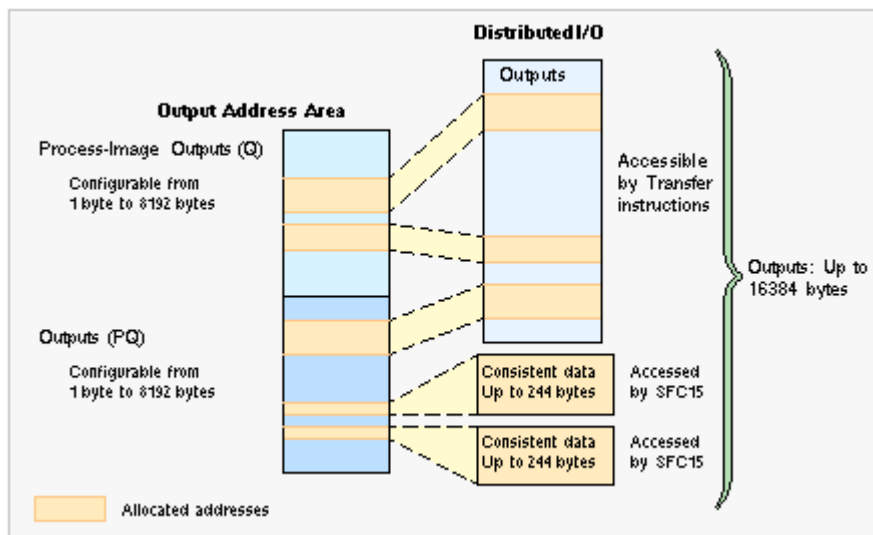
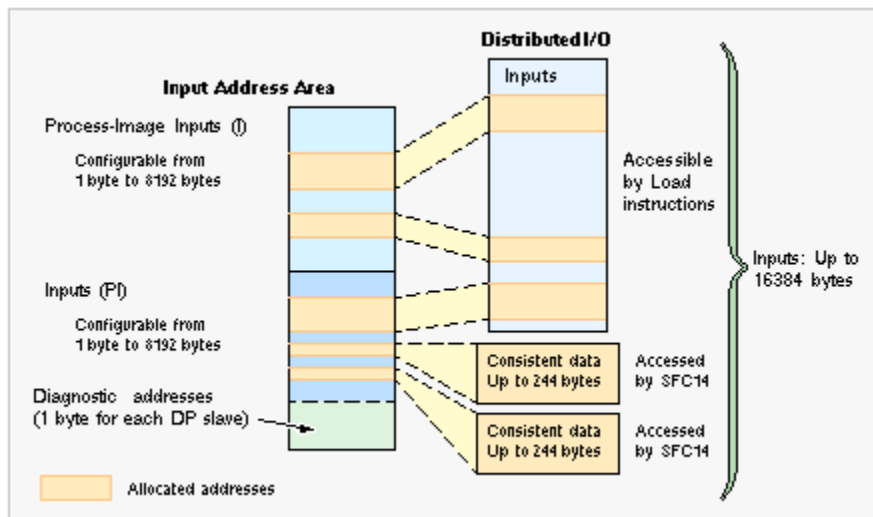
Type of Access	Method
Accessing data in byte, word (2-byte), or double-word (4-byte) units Data integrity is 2 bytes for accessing words, and 4 bytes for accessing double words.	Use the following instructions: <ul style="list-style-type: none"> • The Load instruction reads 1, 2, or 4 bytes of inputs from the I or PI area. • The Transfer instruction writes 1, 2, or 4 bytes of outputs to the Q or PQ area.
Accessing consistent data from a single I/O module in units other than 1 byte, 2 bytes and 4 bytes	Use the following SFCs: <ul style="list-style-type: none"> • SFC14 copies all bytes from the inputs of a module to the I, Q, M, D or L area. • SFC15 writes all bytes from the I, Q, M, D, or L area to the outputs of a module.

As shown in the following figures, your control program can access up to 16384 bytes (each) of inputs and outputs by using the Load (L) and Transfer (T) instructions (for statement list) or the Assign Value (MOVE) instruction (for ladder logic).

Note: You may access any byte of the Process Image (I or Q) area, whether the byte is assigned to physical I/O or not; however, you may only access addresses actually assigned to physical I/O when accessing the Peripheral Image (PI or PQ) or when using SFC14 or SFC15.

SFC14 and SFC15 can access blocks of data up to 244 bytes:

- SFC14 copies the complete block of data from the module's inputs to any of the specified memory areas.
- SFC15 writes the complete block of data from any of the specified memory areas to the module's outputs.



Cross-Module Access Errors

Unlike hardware PLCs, WinLC does not allow a Load (L) or Transfer (T) instruction to access bytes of more than one module. Consider a configuration of two output modules, each containing five bytes. Module 1 is addressed from 10 to 14, and Module 2 is addressed from 15 to 19. OB1 contains the instructions shown below:

```
L 5
T PAW 14
```

In this example, OB122 is called because of an attempt to access bytes across a module boundary. A word instruction at address 14 attempts to access address 14 and 15, which is prevented because the addresses are not in the same module.

Additional Information

For information about the Load (L) and Transfer (T) instructions, see the online help for the STEP 7 programming software and the *Statement List (STL) for S7-300 and S7-400 Programming Reference Manual*. If you are programming in ladder logic, see the Assign Value instruction (MOVE) in the *Ladder Logic (LAD) for S7-300 and S7-400 Programming Reference Manual*.

For information about SFC14 (DPRD_DAT) and SFC15 (DPWR_DAT), see the online help for STEP 7 or the *System Software for S7-300/400 System and Standard Functions Reference Manual*. To view this manual from a computer where STEP 7 is installed, select the **Start > Simatic > Documentation > English** menu command and then double-click "STEP 7 - System and Standard Functions for S7-300 and S7-400".

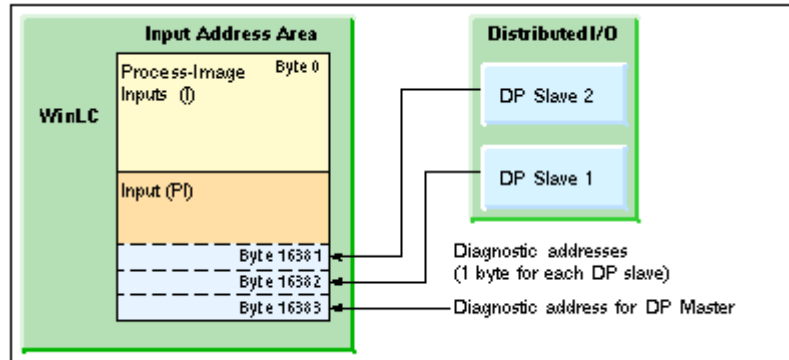
Specifying the Diagnostic Addresses for the DP I/O Modules

During the configuration of WinLC, you allocate a diagnostic address in the peripheral input (PI) area for each node of the DP network. You use the diagnostic address as a parameter for an SFC that accesses the diagnostic data for a node (for example, the LADDR parameter of SFC13). Additionally, this address is used by WinLC to report node state changes (in OB86) to the control program.

Note: STEP 7 documentation sometimes refers to the diagnostic address for a node as the logical base address of the slave or station, as opposed to the logical base address of a module.

As you use the STEP 7 Hardware Configuration tool to configure WinLC and the PROFIBUS-DP network, these diagnostic addresses are assigned above the process-image input (I) memory area. See the figure below.

If you do not enter a specific address, STEP 7 allocates IB16383 for the first DP slave, IB16382 for the second, and so forth.



For more information about configuring the DP diagnostic addresses, see the online help for the STEP 7 programming software.

STEP 7 Components

Logic Blocks Supported by WinLC

Like the other S7 PLCs, WinLC provides several types of logic blocks for processing the user program: organization blocks (OBs), system functions (SFCs), and system function blocks (SFBs). These blocks are an integral part of WinLC.

Organization Block (OB)	System Function (SFC)	System Function Block (SFB)
OB1	SFC0 to SFC6	SFB0 to SFB5
OB10	SFC11 to SFC15	SFB8 and SFB9
OB20	SFC17 to SFC24	SFB12 to SFB15
OB32 to OB36	SFC26 to SFC34	SFB22 and SFB23
OB40	SFC36 to SFC44	SFB32
OB52 to OB57	SFC46 and SFC47	SFB52 to SFB54
OB80, OB82, OB83, OB85, and OB86	SFC49 to SFC52 SFC62 and SFC64	SFB65001 and SFB65002
OB100 and OB102	SFC78 to SFC80	
OB121 and OB122	SFC82 to SFC84 SFC85 and SFC87	

Notes for SFC82, SFC83, and SFC84

In contrast to the S7-300, WinLC supports a synchronous interface for SFC82, SFC83, and SFC84 in STARTUP. WinLC allows both the first call (with REQ = 1) and the second call (with REQ = 0) in STARTUP so the action can be completed in STARTUP.

The normal STEP 7 error codes apply for SFC82, SFC83, and SFC84. In addition, WinLC has a limit of 32 outstanding SFC82, SFC83, and SFC84 jobs. If this limit is exceeded, these SFCs return error code 80C3.

Additional S7 Blocks

In addition to these system blocks, you can use these other S7 blocks to create the control program:

- Function (FC): WinLC supports up to 65,536 FCs (FC0 to FC65535). Each FC can contain up to 65,570 bytes.
- Function block (FB): WinLC supports up to 65,536 FBs (FB0 to FB65535). Each FB can contain up to 65,570 bytes.
- Data block (DB): WinLC supports up to 65,535 DBs (DB1 to DB65535). (DB0 is reserved.) Each DB can contain up to 65,534 bytes.

The number and size of FCs, FBs, and DBs are also limited by the amount of available system memory. For more information about the instruction list supported by WinLC, see the following topics:

Technical data	System functions (SFCs)	Execution times of DP instructions
Organization blocks (OBs)	System function blocks (SFBs)	Execution times of instructions.

Peer-to-Peer Communication Functions

Like other S7 controllers, WinLC provides peer-to-peer communications between controllers on the network. The controllers can be either hardware or software logic controllers.

SFB or SFC	Name	Description
SFB8 SFB9	USEND URCV	Exchange data using a send and a receive SFB
SFB12 SFB13	BSEND BRCV	Exchange blocks of data of variable length between a send SFB and a receive SFB
SFB14 SFB15	GET PUT	Read data from a remote device Write data to a remote device
SFB22 SFB23	STATUS USTATUS	Specific query of the status of a remote device Receive status messages from a remote devices
SFC62	CONTROL	Query the status of a connection

PROFIBUS DPV1

DPV1 extensions to PROFIBUS-DP allow the enhanced communication required by complex slave devices. This enhanced communication includes acyclic data exchange, alarm and status messaging, and the transmission of complex data types. WinLC provides support for the following DPV1 functionality:

- DP-Norm, DP-S7, DPV1, and DPV1 S7-compliant slaves
- Alarm and status OBs for processing DPV1-defined events, including:
 - OB40 (process alarm)
 - OB55 (status alarm)
 - OB56 (update alarm)
 - OB57 (manufacturer-specified alarm)
 - OB82 (diagnostic alarm)
 - OB83 (module pull/plug alarm)
- New data set read and write function blocks:
 - SFB52 (RDREC), Read Data Set
 - SFB53 (WRREC), Write Data Set
 - Execution of SFB54 (RALRM), read alarm data, in the context of the triggering alarm
- Station and interface address
- Buffering of alarms received in DP mode CLEAR

For WinLC to support DPV1, configure the CP card to be a DP Master. To select DP Master, follow these steps from the SIMATIC Manager:

1. Open the Hardware Configuration for your PC Station.
2. Double-click your CP card in the corresponding submodule slot of your WinLC.
3. Select the Operating Mode tab of the CP card Properties dialog.
4. Select DP Master and set the DP mode to DPV1. (DPV1 is the default setting for the CP 5613 card.)

Organization Blocks (OBs)

Organization blocks (OBs) are the interface between the operating system of the controller and the control program. You can use OBs to execute specific components of the user program for the following events:

- When the controller starts and restarts
- Cyclically or at a specific time interval
- At certain times or on certain days
- After running for a specified period of time
- When errors occur
- When a hardware interrupt occurs

The control program in an OB can contain up to 65,570 bytes.

OBs are processed according to the priority assigned to them.

OB	Description	Priority Class
OB1	Main program cycle	1 (lowest)
OB10	Time-of-day interrupt	2 to 24
OB20	Time-delay interrupt	2 to 24
OB32 to OB36	Cyclic interrupt	2 to 24
OB40	Hardware (process alarm) interrupt	2 to 24
OB52 to OB54	ODK interrupt	15
OB55	Status alarm interrupt	2 to 24
OB56	Update alarm interrupt	2 to 24
OB57	Manufacturer-specific alarm interrupt	2 to 24
OB80	Time error	26
OB82	Diagnostic alarm interrupt	24 to 26 (or 28)*
OB83	Module pull/plug alarm interrupt	24 to 26 (or 28)*
OB85	Priority class error	24 to 26 (or 28)*
OB86	Rack (DP slave) failure	24 to 26 (or 28)*
OB100	Warm restart	27
OB102	Cold restart	27
OB121	Programming error	Priority class of the OB where the error occurred
OB122	I/O access error	

* Priority class 28 during STARTUP, user-configurable priority class (from 24 to 26) in RUN mode.

OBs for the Main Program Cycle, Cold Restart, and Warm Restart

The following table shows OBs for the main program cycle and cold and warm restarts. WinLC provides OB1 (main program cycle) for continuously executing the control program. On the transition from STOP mode to RUN mode, WinLC executes OB100 (warm restart) or OB102 (cold restart), based either on the hardware configuration for WinLC or which restart option was selected from a dialog displayed by the WinLC panel. After OB100 (or OB102) has been successfully executed, WinLC executes OB1.

Organization Block (OB)		Start Event (in Hex)	Priority Class
Main program cycle	OB1	1101, 1103, 1104	1
Warm restart	OB100	1381, 1382	27
Cold restart	OB102	1385, 1386	27

Interrupt OBs

WinLC provides a variety of OBs that interrupt the execution of OB1. The following table lists the different interrupt OBs that are supported by WinLC. These interrupts occur according to the type and configuration of the OB.

The priority class determines whether the controller suspends the execution of the control program (or other OB) and executes the interrupting OB. You can change the priority class for the interrupt OBs.

Interrupts		Start Event (in Hex)	Default Priority Class
Time-of-Day Interrupt	OB10	1111 (OB10)	2
Time-Delay Interrupt Range: 1 ms to 60000 ms	OB20	1101 (OB20)	3
Cyclic Interrupt Range: 1 ms to 60000 ms Recommended: > 10 ms	OB32	1133	9
	OB33	1134	10
	OB34	1135	11
	OB35	1136	12
	OB36	1137	13
Hardware interrupt	OB40	1141 (channel1)	16
Status Alarm interrupt	OB55	1155	2
Update Alarm interrupt	OB56	1156	2
Manufacturer-Specific Alarm interrupt	OB57	1157	2

If WinLC has been configured to execute a particular interrupt OB, but that OB has not been downloaded, WinLC reacts in the following manner:

- If OB10, OB20, OB40, OB55, OB56, or OB57 is missing and OB85 has not been downloaded, WinLC changes operating mode (from RUN to STOP).
- WinLC remains in RUN mode if a cyclic interrupt OB (OB32 to OB36) is missing. If these OBs cannot be executed at the specified time and OB80 has not been downloaded, WinLC changes from RUN mode to STOP mode.

Note: If you schedule a cyclic interrupt OB (OB32 to OB36) to be executed at a specific interval, make certain that the program can be executed within the time frame and also that your control program can process the OB within the allotted time.

Error OBs

WinLC provides a variety of error OBs. Some of these error OBs have the configured (the user-assigned) priority class, while others (OB121 and OB122) inherit the priority class of the block where the error occurred.

The local variables for OB121 and OB122 contain the following information that can be used by the control program to respond to the error:

- The type of block (byte 4) and the number (bytes 8 and 9) where the error occurred
- The address within the block (bytes 10 and 11) where the error occurred

If the start event occurs for a particular error OB that has not been downloaded, WinLC changes operating mode from RUN to STOP.

Error or Fault		Start Event (in Hex)	Default Priority Class
Time error	OB80	3501, 3502, 3505, 3507	26
Diagnostic Interrupt	OB82	3842, 3942	26
Insert/remove module interrupt	OB83	3861, 3863, 3864, 3865, 3961	26
Priority class error: <ul style="list-style-type: none"> • Start event occurs for an OB that has not been downloaded. • During the I/O cycle, WinLC attempts to access a module or DP slave that is defective or not plugged in. • WinLC attempts to access a block (such as a DB) that has not been downloaded or has been deleted. 	OB85	<ul style="list-style-type: none"> • 35A1 • 39B1, 39B2, 39B3/38B3, 39B4/38B4 (depending on the OB85 Call Up method) • 35A3 	26
Rack failure (distributed I/O): a node in the PROFIBUS-DP subnet has failed or has been restored.	OB86	38C4, 38C5, 38C7, 38C8, 39C4, 39C5	26 (or 28)
Programming error (For example: the user program attempts to address a timer that does not exist.)	OB121	2521, 2522, 2523, 2524, 2525, 2526, 2527, 2528, 2529, 2530, 2531, 2532, 2533, 2534, 2535, 253A; 253C, 253E	Same priority class as the OB in which the error occurred
I/O access error (For example: the user program attempts to access a module that is defective or is not plugged in.)	OB122	2942, 2943	

System Functions (SFCs)

WinLC provides SFCs, which are system functions that perform various tasks. The control program calls the SFC and passes the required parameters; the SFC performs its task and returns the result.

- WinLC allows a maximum of 5 instances of the asynchronous system function SFC51 (index B1, B3) to be running.
- WinLC allows a maximum of 20 asynchronous SFCs from the following set to be running: SFC11, SFC13, SFC55, SFC56, SFC57, SFC58, and SFC59.
- WinLC allows a maximum of 32 asynchronous SFCs in any combination from the following set to be running: SFC82, SFC83, and SFC84.

For more information on these functions, see the online help for STEP 7 or the *System Software for S7-300/400 System and Standard Functions Reference Manual*. To view this manual from a computer where STEP 7 is installed, select the **Start > Simatic > Documentation > English** menu command and then double-click "STEP 7 - System and Standard Functions for S7-300 and S7-400".

SFCs That Can Cause the Scan Cycle to Vary

The following SFCs can cause the scan cycle to vary ("jitter"):

- SFC22 (CREAT_DB)
- SFC23 (DEL_DB)
- SFC52 (WR_USMG)
- SFC85 (CREA_DB)

Execution Times

The following table lists the execution times for the SFCs supported by WinLC. Execution times for asynchronous SFCs refer to the time for the SFC call, not the time for the job to complete.

Note: The execution times were measured on a SINUMERIK PCU50 computer (with a single 1.2 GHz Celeron processor). Tuning settings: 9000 microsecond execution time limit, 90% maximum execution load, and 1000 μ s forced execution sleep. Actual execution times may vary, depending on your computer.

SFC	Name	Description	Execution Time
SFC0	SET_CLK	Sets the system clock.	1.55 μ s
SFC1	READ_CLK	Reads the system clock.	1.46 μ s
SFC2	SET_RTM	Sets the run-time meter.	0.53 μ s
SFC3	CTRL_RTM	Starts or stops the run-time meter.	0.53 μ s
SFC4	READ_RTM	Reads the run-time meter.	0.43 μ s
SFC5	GADR_LGC	Queries the logical address of a channel.	1.95 μ s
SFC6	RD_SINFO	Reads the start information of an OB.	1.67 μ s
SFC11	DPSYNC_FR	Synchronizes groups of DP slaves.	1.12 μ s

SFC	Name	Description	Execution Time
SFC12	D_ACT_DP	Deactivates and activates of DP slaves.	—
SFC13	DPNRM_DG	Reads the diagnostic data of a DP slave. DP configuration tested: one ET 200M slave with one 8-input/8-output module and one 16-output module	2.20 μ s
SFC14	DPRD_DAT	Reads the consistent data from a DP slave.	1.71 μ s
SFC15	DPWR_DAT	Writes the consistent data to a DP slave.	1.87 μ s
SFC17	ALARM_SQ	Generates an acknowledgeable block-related message.	4.64 μ s
SFC18	ALARM_S	Generates an unacknowledgeable block-related message.	4.61 μ s
SFC19	ALARM_SC	Queries the status for the last message (SFC17 or SFC18).	1.23 μ s
SFC20	BLKMOVB	Copies variables.	1.88 μ s
SFC21	FILL	Initializes a memory area: 1 word 50 words 100 words	1.81 μ s 2.82 μ s 3.83 μ s
SFC22	CREAT_DB	Creates a retentive data block in Work memory. The current values of the DB are retained after a warm restart.	14.20 μ s
SFC23	DEL_DB	Deletes a data block. WinLC allows an application to delete a non-sequence-relevant data block.	4.73 μ s
SFC24	TEST_DB	Provides information about a data block. For WinLC, SFC24 can return the DB length and write-protection flags for non-sequence-relevant data blocks, although it returns error code 80B2 for non-sequence-relevant data blocks.	0.63 μ s

SFC	Name	Description	Execution Time
SFC26	UPDAT_PI	Updates the process-image input table. DP configuration tested: one ET 200M slave with one 8-input/8-output module and one 16-output module	0.83 μ s
SFC27	UPDAT_PO	Updates the process-image output table.	16.30 μ s
SFC28	SET_TINT	Sets the time-of-day interrupt (OB10).	2.05 μ s
SFC29	CAN_TINT	Cancel the time-of-day interrupt (OB10).	0.74 μ s
SFC30	ACT_TINT	Activates the time-of-day interrupt (OB10).	0.63 μ s
SFC31	QRY_TINT	Queries the time-of-day interrupt (OB10).	0.80 μ s
SFC32	SRT_DINT	Starts the time-delay interrupt (OB20).	2.26 μ s
SFC33	CAN_DINT	Cancel the time-delay interrupt (OB20).	0.82 μ s
SFC34	QRY_DINT	Queries the time-delay interrupt (OB20).	0.80 μ s
SFC36	MSK_FLT	Masks synchronous errors.	0.72 μ s
SFC37	DMSK_FLT	Unmasks synchronous errors.	0.72 μ s
SFC38	READ_ERR	Reads the error register.	0.71 μ s
SFC39	DIS_IRT	Disables the processing of all new interrupts.	0.70 μ s
SFC40	EN_IRT	Enables the processing of new interrupts.	0.71 μ s
SFC41	DIS_AIRT	Disables the processing of new interrupts with higher priority than the current OB.	0.36 μ s
SFC42	EN_AIRT	Enables the processing of new interrupts with higher priority than the current OB.	0.86 μ s
SFC43	RE_TRIGR	Retriggers the watchdog timer (monitoring the cycle time).	4.49 μ s
SFC44	REPL_VAL	Transfers a value to ACCU1 (accumulator 1).	15.18 μ s
SFC46	STP	Changes the operating mode to STOP mode.	—

SFC	Name	Description	Execution Time
SFC47	WAIT	Delays the execution of the control program by the specified number of microseconds, rounded up to the nearest millisecond.	—
SFC49	LGC_GADR	Queries the module slot belonging to a logical address.	1.40 μ s
SFC50	RD_LGADR	Queries all of the logical addresses of a module	3.08 μ s
SFC51	RDSYSST	Reads all or part of a system status list.	7.81 μ s
SFC52	WR_UMSG	Writes a user element to the diagnostics buffer.	4.74 μ s
SFC54	RD_PARM	Reads the defined parameter.	2.02 μ s
SFC55	WR_PARM	Writes the defined parameter.	2.35 μ s
SFC56	WR_DPARM	Writes the default parameter.	1.44 μ s
SFC57	PARAM_MOD	Assigns the parameters to a module.	1.24 μ s
SFC58	WR_REC	Writes a data record.	2.66 μ s
SFC59	RD_REC	Reads a data record.	2.61 μ s
SFC62	CONTROL	Checks the status of an SFB instance.	—
SFC64	TIME_TCK	Reads the time from the system clock.	0.42 μ s
SFC78	OB_RT	Reports OB run-time information, with resolution to the nearest millisecond.	—
SFC79	SET	Sets a range of outputs.	0.86 μ s
SFC80	RESET	Resets a range of outputs.	0.81 μ s
SFC82	CREA_DBL	Creates a data block in Load memory.	—
SFC83	READ_DBL	Copies data data from a block in Load memory.	—
SFC84	WRIT_DBL	Writes to a Load Memory block so that the data is saved immediately. Load memory blocks that are used to recover from an abnormal termination can be updated while the program is running. Use SFC84 only for larger segments of a database, not for frequent variable processing.	—

SFC	Name	Description	Execution Time
SFC85	CREA_DB	Creates a DB that can be either retentive or non-retentive, depending on the input parameter: <ul style="list-style-type: none">• If retentive, the current values of the DB are retained after a warm restart (OB100).• If non-retentive, the current values of the DB are not retained after a warm restart (OB100).	—
SFC87	C_DIAG	Determines the current status of all S7 connections.	—

System Function Blocks (SFBs)

System Function Blocks are logic blocks (similar to SFCs) that perform basic tasks when called by the control program. The following table lists the execution times of the SFBs that are supported by WinLC. You must provide a data block (DB) when you call an SFB.

For more information on these functions, see the online help for STEP 7 or the *System Software for S7-300/400 System and Standard Functions Reference Manual*. To view this manual from a computer where STEP 7 is installed, select the **Start > Simatic > Documentation > English** menu command and then double-click "STEP 7 - System and Standard Functions for S7-300 and S7-400".

Note: The execution times were measured on a SINUMERIK PCU50 computer (with a single 1.2 GHz Celeron processor). Tuning settings: 9000 microsecond execution time limit, 90% maximum execution load, and 1000 μ s forced execution sleep. Actual execution times may vary, depending on your computer.

SFB	Name	Description	Execution Time
SFB0	CTU	Provides a count-up counter	0.03 μ s
SFB1	CTD	Provides a count-down counter	0.03 μ s
SFB2	CTUD	Provides a count-up/down counter	0.04 μ s
SFB3	TP	Generates a pulse	0.04 μ s
SFB4	TON	Generates an on-delay timer	0.04 μ s
SFB5	TOF	Generates an off-delay timer	0.04 μ s
SFB8	USEND	Sends a data packet of CPU-specific length (two-way)	—
SFB9	URCV	Receives a data packet of CPU-specific length (two-way)	—
SFB12	BSEND	Sends a data block up to 64 Kbytes (two-way)	—
SFB13	BRCV	Receives a data block up to 64 Kbytes (two-way)	—
SFB14	GET	Reads data up to a CPU-specific maximum length (one-way)	—
SFB15	PUT	Writes data up to a CPU-specific maximum length (one-way)	—
SFB22	STATUS	Query the status of a remote device	—
SFB23	USTATUS	Receive the status of a remote device	—

SFB	Name	Description	Execution Time
SFB32	DRUM	Implements a sequencer	0.15 μ s
SFB52	RDREC	Read data set	—
SFB53	WRREC	Write data set	—
SFB54	RALRM	Read alarm data for a DP slave	—
SFB65001	CREA_COM	(WinAC ODK)	—
SFB65002	EXEC_COM	(WinAC ODK)	—

System Clock and Run-Time Meter

Like a hardware S7 controller, WinLC has a "real-time" system clock (based on the hardware clock of your computer).

Note: Setting the time for WinLC does not affect the time setting for your computer.

To adjust and read the system clock, use the following SFCs. For more information on these functions, see the online help for STEP 7 or the *System Software for S7-300/400 System and Standard Functions Reference Manual*. To view this manual from a computer where STEP 7 is installed, select the **Start > Simatic > Documentation > English** menu command and then double-click "STEP 7 - System and Standard Functions for S7-300 and S7-400".

SFC	Name	Description
SFC0	SET_CLK	Allows you to set the time and the date of the system clock. The clock then runs starting from the set time and set date. Format: DT#1995-01-15-10:30:30
SFC1	READ_CLK	Allows you to read the current date or current time of the system clock of the controller.
SFC64	TIME_TCK	Allows you to read the system time of the controller. The system time is a "time counter" counting cyclically from 0 to a maximum of 2147483647 ms. In case of an overflow, the system time is counted again, starting at 0. The time base (and therefore the accuracy) is 1 ms. The system time is only influenced by the operating modes of the controller.

Tuning the Performance of the Controller

Scan Cycle for a PC-Based Controller

During one scan cycle, the controller updates the outputs, reads the inputs, executes the control program, performs communication tasks, and provides time for other applications to run. The following parameters affect the scan cycle:

- Execution Time (in milliseconds) is the actual amount of time used by the controller to update the I/O and to execute the control program.
- Scan Cycle Time (in milliseconds) is the number of milliseconds from the start of one cycle to the start of the next cycle. This value must be greater than the execution time of the scan to provide execution time for any application that has a lower priority than WinLC.
- Sleep Time (in milliseconds) determines how much time is available during the free cycle (execution cycle for OB1) to allow higher priority OBs and other applications to use the resources of the computer.

The Priority for the controller application also affects the scan cycle by determining when the controller runs or is interrupted by other Windows applications. If the priority for the controller is set to the Normal priority (priority level 8) or above, you must ensure that the sleep time occurs every 50 milliseconds or less in order for other Windows applications, such as moving the mouse, to operate smoothly.

The tuning panel allows you to tune and test the performance of the controller by adjusting the parameters that affect the scan cycle (minimum cycle time, minimum sleep time, and priority) without affecting the system configuration for the controller. After testing tuning parameters, you use STEP 7 to configure the minimum scan cycle time for the controller when you create the system (hardware) configuration.

Tasks Performed during the Scan Cycle

After you have used STEP 7 to create and download your control program to the controller, the controller starts executing the control program when you set the controller to RUN or RUN-P mode. Like any other S7 PLC, the controller executes your control program in a continuously repeated scan cycle.

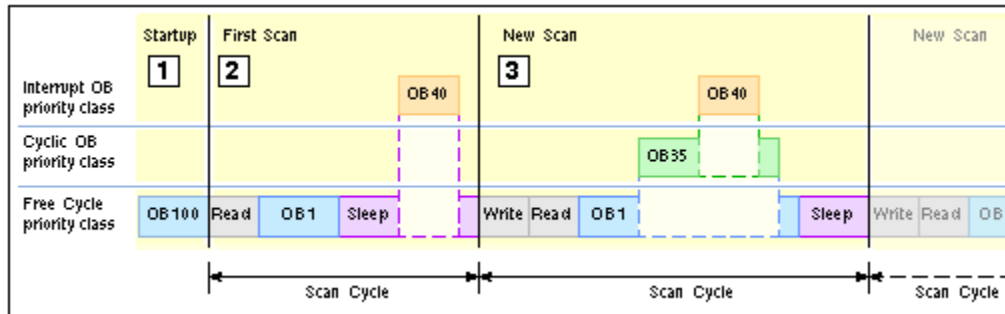
In one scan, the controller performs the following tasks:

1.	The controller writes the status of the OB1-assigned process-image output table (the Q memory area) to the I/O module outputs. For the first scan, the controller does not perform this task.
2.	The controller reads the states of the I/O module inputs into the OB1-assigned process-image input table (the I memory area).
3.	The controller executes the control program in OB1.
4.	OB1 waits until the minimum sleep time requirement is met before starting another scan. Other OBs can execute at this time.

STEP 7 Operations

Because the PC-based controller shares the resources of your computer with other programs (including the operating system), you must ensure that the controller provides sufficient time for other Windows applications to be processed. If the actual execution time for the scan cycle is less than the minimum scan cycle time that you configured with STEP 7, the controller suspends the free cycle (OB1) until the minimum scan cycle time is reached before starting the next scan. This waiting period, or sleep time, allows other applications to use the resources of the computer.

The following figure provides an overview of the tasks that are performed by the controller during different scan cycles.



1 Startup On a transition from STOP mode to RUN mode, the controller loads the system configuration, sets the I/O to the default states, and executes the startup OB (OB100 or OB102).

The startup cycle is not affected by the minimum cycle time and minimum sleep time or watchdog parameters; however, it is affected by the execution time limit.

2 First Scan The first scan following the startup OB does not write to the outputs, but starts by reading the inputs. An OB with a higher priority class can interrupt the free cycle at any time, even during the sleep time.

In the example above, the controller handles a hardware (I/O) interrupt that occurs during the sleep time by executing OB40. After OB40 has finished, the controller waits for the minimum cycle time to start the next scan.

Note: It is possible for the controller to use all of the sleep time for processing higher-priority OBs. In this case, other Windows applications may not have sufficient time to run. Refer to the techniques for managing sleep time listed below.

3 New Scan The new scan starts by writing to the outputs. In the example above, the controller suspends the execution of OB1 to execute a cyclic OB (OB35), which has a higher S7 priority than OB1. The controller also suspends the execution of OB35 to handle another I/O interrupt (OB40).

After OB40 finishes, the controller resumes the execution of OB35, and after OB35 finishes, the controller resumes the execution of OB1.

The length of the scan cycle is determined by the execution time of all OBs executed during the scan, the minimum cycle time, and the minimum sleep time. If the execution time is less than the minimum cycle time that was configured in the system configuration, the controller suspends the free cycle until the minimum sleep time is met. During the sleep time, the computer runs any interrupt OBs and other Windows applications.



Warning

Variation in the execution time or response time of the control program could potentially create a situation where the equipment or application being controlled can operate erratically and possibly cause damage to equipment or injury to personnel.

If the controller does not provide sufficient sleep time for the other applications to run, the computer can become unresponsive to operator input, or the controller and other applications can operate erratically. In addition, the execution of the control program can experience non-deterministic behavior (jitter) such that execution times can vary and start events can be delayed.

Always provide an external emergency stop circuit. In addition, always tune the sleep time and manage the performance of the controller so that your control program executes consistently.

Methods for Managing the Performance of WinLC

While executing the control program, WinLC can experience a variation in the process execution time or response time that causes the scan times to vary or to exhibit non-deterministic behavior ("jitter"). You can use the following methods to manage the performance of WinLC:

- Adjusting the priority for the controller: Affects when the operating system runs or interrupts the controller
- Adjusting the minimum sleep time and minimum cycle time parameters: Affects the execution of the free cycle or OB1 (OB priority class 1)
- Inserting sleep time into the control program (SFC47 "WAIT"): Affects the execution for the priority class of the OB that calls SFC47 (and any lower priority class)
- Adjusting the sleep-monitoring algorithm of the execution monitor: Affects the execution of all OB priority classes (if the other mechanisms do not meet the requirements for sleep time)

WinLC provides a tuning panel for monitoring the performance and for modifying the parameters that affect the scan cycle.

What Causes Jitter?

Because the PC-based controller must share the computer with other running processes, the execution of the control program can experience "jitter" when a higher-priority or active process uses the CPU or system resources of the computer. Jitter is a variation in the process execution time or response time that causes the scan times to vary or to exhibit non-deterministic behavior.

Jitter occurs when there is a delay in either the start or the finish of an OB. For example: the execution time can deviate by a few milliseconds between scans, or the start of an interrupt OB can be delayed. For some control applications, such time lapses do not disturb the proper operation of the controller, but in a highly time-sensitive process, a jitter of even 1 ms can be significant.

The following settings for WinLC can cause jitter in the execution of the control programs:

- Priority settings for different applications
- Priorities among the WinLC threads
- Execution Monitor (that measures the CPU usage)

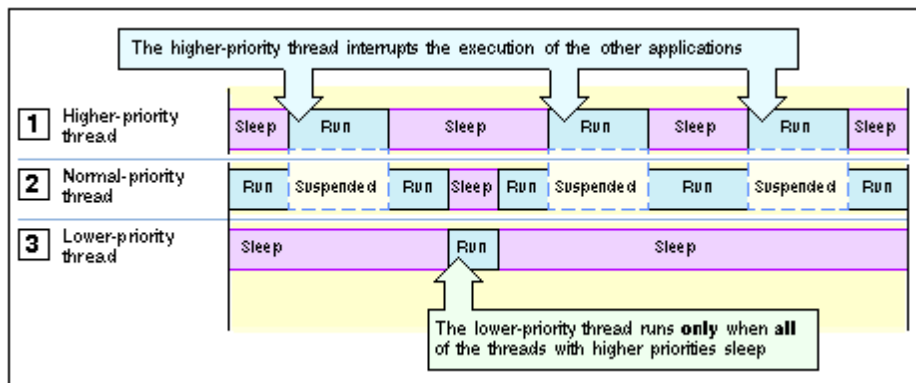
The tuning panel of WinLC provides several tools for reducing jitter in the control program.

Jitter can also be caused by other sources than WinLC:

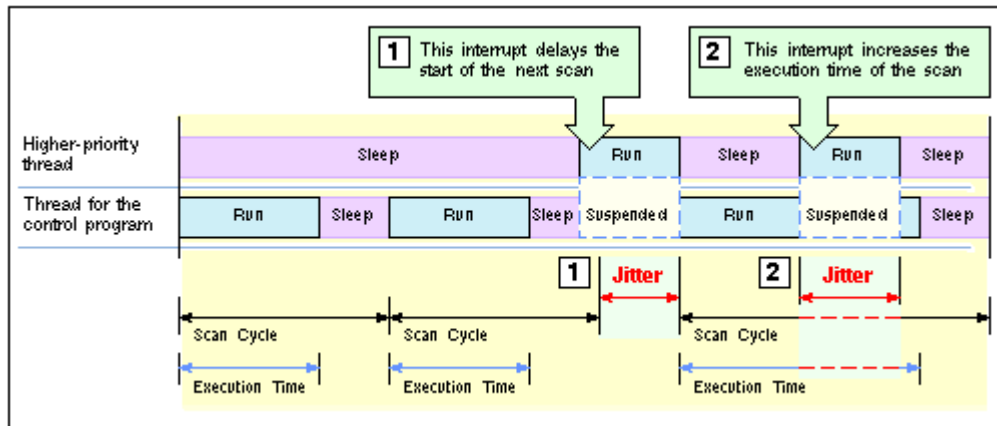
- Jitter can be caused by the design of your control program. For example, different branches in the logic of the control program might cause the execution time to vary.
- Jitter can be caused by the computer hardware. For example, jitter can be caused by an operation with a long DMA cycle, such as a video card using the PCI bus. Jitter can also be caused by a driver, such as for a CD drive or a diskette drive. Hardware-induced jitter cannot be managed by software.
- Jitter can be caused by an application that was created with the WinAC Open Development Kit (ODK), such as when a synchronous process takes too long to execute. Refer to the documentation for WinAC ODK for more information.

Priority Settings for Different Applications Can Cause Jitter

Every application that is running on your computer has one or more threads (or tasks), and each thread has a priority. The Windows operating system executes the threads with the highest priority first and executes a lower-priority thread only when all of the higher priority threads are finished or suspended (for example, to wait for some other activity to complete or to "sleep" for a specified time). Threads with higher priorities interrupt and suspend the operations of threads with lower priorities. After the higher-priority thread finishes, the lower-priority thread resumes its operation.



Jitter can occur when a process of an application with a higher priority interrupts and suspends the execution of the controller. As shown in the following figure, jitter typically appears in two forms.



- 1 The higher priority threads can cause jitter by delaying the start of an OB. This could delay the start of the free cycle (OB1) or of an interrupt OB (such as OB35 or OB40).
- 2 The higher priority application can cause jitter by extending the execution time for an individual scan.

You can use the tuning panel to increase or decrease the priority for the WinLC threads. The higher you set the priority for the WinLC threads in relation to the threads of the other applications, the less jitter you typically encounter. However, you must also ensure that WinLC provides enough sleep time for the other applications to run.

The tuning panel also provides information that allows you to monitor the amount of jitter in the scan cycle.

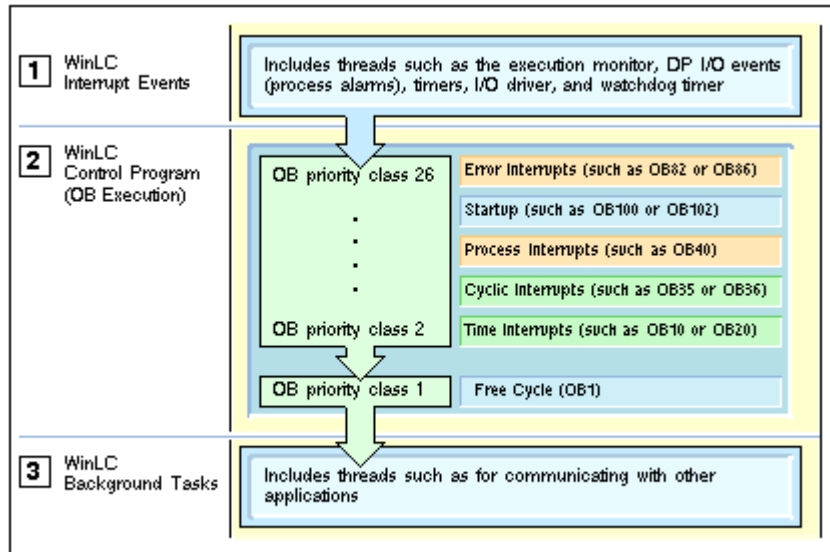
For more information about priorities, refer to the following topics:

- Adjusting the Priority
- Windows Priorities

Priorities among the WinLC Threads Can Cause Jitter

In addition to the thread that executes the OBs of the control program, WinLC uses other threads, including some with higher priority than the OB Execution thread. Some examples of higher-priority threads are the execution monitor, the start event for an OB, the watchdog events, the timers, the PROFIBUS-DP driver, and the events for the DP I/O. Any of these higher-priority threads can induce jitter in the execution of the control program.

The relative priorities (priority classes) of the OBs in the control program itself can also cause jitter. For example, an error OB delays or interrupts the execution of all lower-priority OBs.

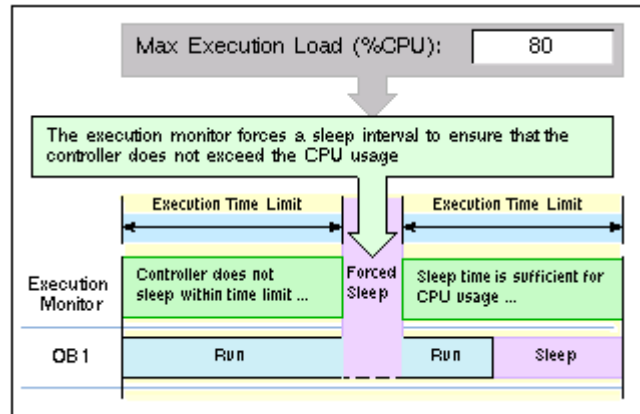


- 1** The threads of the interrupt events have a higher priority than the thread for execution the control program. These threads can cause jitter by interrupting the execution of the control program.
- 2** The OB Execution thread includes the different priority classes for the OBs of the control program. The interrupt OBs can cause jitter not only by interrupting the free cycle (OB1), but also by interrupting another interrupt OB with a lower priority class.
- 3** The background tasks for WinLC includes the threads used for communicating with other applications, such as STEP 7. The OB Execution thread and the higher-priority threads affect the execution of these tasks.

The Sleep Interval Forced by the Execution Monitor Can Cause Jitter

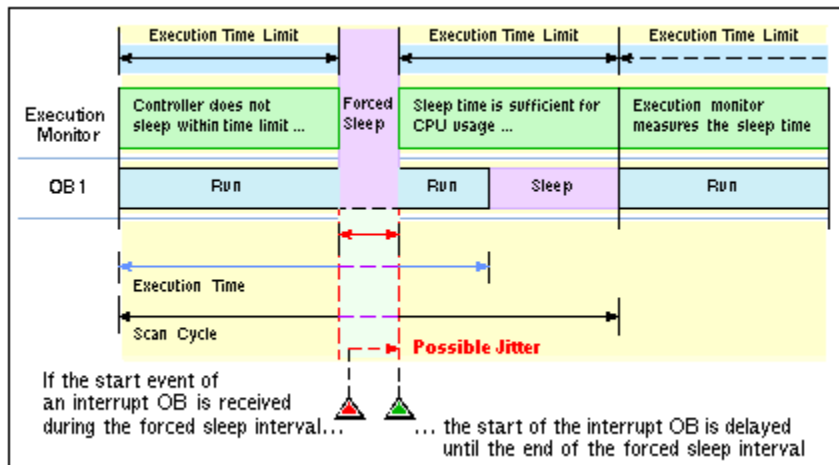
WinLC must sleep (release the CPU) periodically in order for the other applications to run. The free cycle includes a sleep interval that follows the execution of OB1. However, this sleep interval can be interrupted by higher-priority OBs. Also, a scan cycle with a relatively long execution time could cause other applications to wait too long to access the CPU.

To ensure that the controller does not exceed a specified percentage of CPU usage, an execution monitor measures the sleep time within a fixed execution time limit. If the controller does not sleep for the specified amount of time within the execution time limit, the execution monitor forces a sleep interval.



Because the execution monitor runs in a higher priority class than any OB, the controller cannot interrupt the forced sleep interval. This could delay the start of an interrupt OB, such as OB35, until the end of the forced sleep interval. This delay in handling an interrupt OB results in jitter.

As a general rule for decreasing jitter, always design your control program to keep the execution time of the higher-priority OBs as short as possible.



WinLC provides several options for managing the sleep time to avoid the uninterruptible forced sleep interval:

- You can increase the minimum sleep time parameter for managing the sleep time for the free cycle (priority class 1, or OB1).
- You can call SFC47 (“WAIT”) to insert an extra, interruptible sleep interval into the control program for managing the sleep time for an application-defined priority class (priority classes 2 to 24).
- You can adjust the sleep-monitoring algorithm for the execution monitor for managing sleep time at a higher priority class than any OB.

Adjusting the Priority of the Controller

The priority of the controller determines how WinLC runs in relation to other Windows applications that are running on the computer.

Adjusting the priority of the controller can reduce or increase the amount of jitter in the scan time. The tuning panel allows you to change the priority for the controller application. When you use the tuning panel to change the priority, the controller automatically ensures that its interrupt activities, such as those which schedule interrupt OBs, are also set to an appropriate priority.

WinLC does not control priorities in customer software, such as asynchronous threads in custom software or other applications in the same environment.

Note: The CCX interface of the WinAC Open Development Kit (ODK) provides an ODK_CreateThread function. Calling the ODK_CreateThread function creates asynchronous threads with priorities that are adjusted when you change the priority of the controller.

If you do **not** use the ODK_CreateThread function to create threads (for example, if you use a Windows API call to create a thread), changing the priority of the controller does **not** adjust the priority of those threads.

Refer to the documentation of the WinAC Open Development Kit (ODK) for more information.

While a PC-based controller must maintain the essential features of a SIMATIC S7 PLC, the PC-based controller must also allow the other applications to run on the computer. The operating system of the computer uses a concept of execution threads (or tasks) to run or execute the applications. Each application has one or more threads, and each thread has a priority. The operating system executes the threads with the highest priority first and executes a lower-priority thread only when all of the higher priority threads are suspended (for example, to wait for some other activity to complete or to “sleep” for a specified time). Threads with higher priorities interrupt and suspend the operations of other threads that have lower priorities. After the higher-priority thread finishes, the lower-priority thread resumes its operation.

To change the priority, follow these steps:

1. Use the Priority slider to choose a priority based on the priority levels for your operating system. The new priority is displayed as you move the slider.
2. Click Set to set the priority to the new value.

Windows Priorities

The Windows operating system groups priorities into four groups: Normal, Above Normal, Real-Time, or Background. Adjusting the priority of the WinLC application determines whether the other applications running on the computer can interrupt WinLC.

Priority of the Controller	Effect on the Scan Cycle
<p>Same or lower priority to other Windows applications:</p> <ul style="list-style-type: none"> • Normal (8) • Background (1 to 7) 	<p>The operating system of the computer determines the scheduling of actual start and completion time for the scan cycle. The scan cycle and the execution of the control program can become non-deterministic (and can increase the amount of jitter).</p> <p>Because the controller runs only when other applications have finished, the lower priority allows the controller access more of the functions of the CPU (such as for mathematical processing). However, waiting for other applications to finish can lead to scan times that vary greatly.</p>
<p>Higher priority than other Windows applications:</p> <ul style="list-style-type: none"> • Above Normal (9 to 13) • Real-Time (17 to 29) 	<p>The controller plays a greater role in determining when to allow time for the other Windows applications to run. This yields a more deterministic scan time and reduces jitter.</p> <p>However, there is a possibility that the controller may not allow enough time for the other applications to run, which could then cause the other applications to lock up or become disabled.</p> <p>To ensure that the other Windows applications have enough time to run, you must tune the performance of the controller by adjusting the minimum cycle time and minimum sleep time parameters.</p>

You use the Priority slider on the tuning panel of WinLC to adjust the priority for the WinLC application. If you attempt to set the priority to a level that is not supported by WinLC, the tuning panel automatically sets the priority to the next lowest priority that is supported by WinLC.

Normal Priority (8)

The default priority for the controller is Normal (8), which is the default priority for all Windows applications. For a control project that is not time-critical and includes intensive calculations that require extensive use of the CPU resources of the computer, consider using the Normal priority.

Do **not** use the Normal priority for a control application that is time-critical.

Because most Windows applications run at the Normal priority, executing the controller at the Normal priority helps to ensure that the other applications have sufficient access to the resources of the computer.

However, the Normal setting also gives the operating system more control of when the scan cycle starts and finishes. Shifting the control for starting and stopping the scan cycle to the operating system induces the potential of more non-deterministic behavior, or jitter, for the scan cycle.

Background Priority (1 to 7)

For a control project that is not time-critical and includes intensive calculations that require extensive use of the CPU resources of the computer, consider using a Background priority. Background priority applications are at a lower priority than Normal priority applications.

Do **not** use a Background priority for a control application that is time-critical.

Setting the priority from 1 to 7 configures the controller as a background task that executes whenever the higher-priority applications are not using the resources of the computer. The applications with higher priority operate smoothly, even if you have not configured the controller for a minimum sleep time.

Setting the priority to 1 defines an idle process that executes only when the computer has no other task to execute.

Selecting a background priority allows the controller to maximize the use of the CPU resources of the computer. Setting the minimum sleep times to 0 ensures that the controller maximum access to the CPU resources. However, a scan cycle for a controller set to a background priority may vary considerably (have jitter, or be non-deterministic), depending on how much execution time the other applications require.

Above-Normal Priority (9 to 13)

For a control project that is less time-critical but requires a more deterministic scan cycle, consider using an Above-Normal priority. This level of priority reduces the amount of jitter in the scan cycle.

You should provide sleep time that allows other application to run. (For more information about managing sleep time, see the topic on Sleep Management Techniques.) Use the tuning panel to monitor the variation in scan times that occurs as the controller executes your control program.

Setting the priority from 9 to 13 configures the controller to run with a higher priority than typical Windows applications. (However, other Windows applications often compete in this level of priority.) The Above-Normal priority allows the controller to maintain a more deterministic scan, but the operating system still interrupts the controller operations periodically.

For a controller set to run at Above-Normal priority, you must be more careful about the CPU usage for the computer. If the controller does not sleep periodically (giving access to the CPU), other Windows applications can be severely affected.

Be aware that the Windows operating system also periodically increases the priority for any lower-priority applications to ensure that these applications get some amount of CPU time. This could allow some lower-priority applications, in addition to higher-priority applications, to create jitter in the scan cycle.

Windows "Real-Time" Priority (17 to 29)

For control projects that are very time-critical, consider using a Real-Time priority. With this level of priority, the scan cycle is very deterministic, with a minimal amount of jitter in the scan cycle. Even at this level of priority, there are still some Windows events that can interrupt the execution of the controller.

You should provide sleep time that allows other application to run. (For more information about managing sleep time, see the topic on Sleep Management Techniques.) Use the tuning panel to monitor the variation in scan times that occurs as the controller executes your control program.

With a Real-Time priority, the controller typically competes only with high-priority applications (such as those that control the mouse), drivers, special Windows features (such as delayed procedure calls or DPCs), and some features of the hardware.

This priority range allows WinLC to have a much more deterministic scan cycle. Typical jitter of less than 10 milliseconds is possible, depending on the hardware platform and other software installed.

Managing the Sleep Time

Sleep Management Techniques

During a sleep interval, the controller allows other applications to use the resources of the computer. By managing the sleep time, you can tune the performance of the controller in order to allow all applications on the computer to run with acceptable performance. You can use a variety of techniques for managing the sleep intervals for the controller:

- Adjusting the minimum sleep time parameter. The minimum sleep time determines the amount of sleep time that is added during the execution of the free cycle (OB1). This sleep time affects only OB priority class 1.
- Calling SFC47 from your control program. SFC47 inserts a sleep interval into the execution of your control program. This sleep time affects OB priority classes 2 to 24.
- Adjusting the execution monitor. The execution monitor uses a sleep-monitoring algorithm (based on the execution time limit and the maximum execution load parameters) to force a sleep interval. The execution monitor runs asynchronously to the scan cycle. This sleep time affects all OB priority classes.

Contents of this topic:

- Managing the Sleep Time of the Controller
- Tuning Strategy
- Sample Interaction of the Execution Monitor and the Minimum Sleep Time

Managing the Sleep Time of the Controller

Because the controller shares the resources of your computer with other applications, you must ensure that the controller sleeps for a sufficient interval to allow the other applications to run.

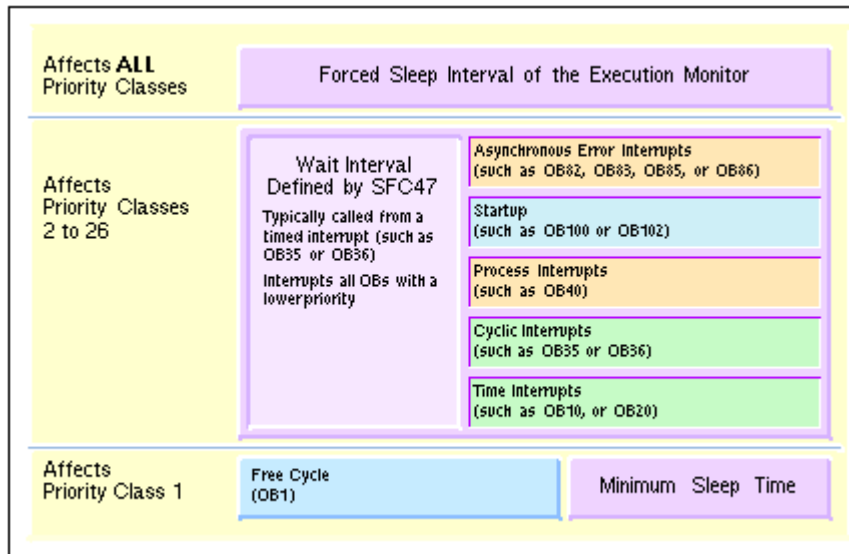
Notice

The most effective method for granting time to other applications is to set the minimum sleep time parameter to the largest value that your control application allows. The other methods for managing the sleep time provide sufficient sleep time for the other applications to run, but may degrade the performance of the controller.

The controller provides the following techniques for managing the sleep time:

- The controller provides an execution monitor that enforces the maximum execution load on the resources of the computer. The execution monitor measures the amount of sleep time taken by the controller within an execution time limit, which is independent from the execution time of the scan cycle. If necessary, the execution monitor forces a sleep interval to achieve the specified execution load. This forced sleep interval suspends the execution of any OB and can also delay the start of an interrupt OB.
- The controller provides a minimum sleep time parameter that adds sleep time for the free cycle. This sleep interval occurs after the execution of OB1. The minimum sleep time affects only priority class 1. An OB in a higher priority class can interrupt this sleep interval. The controller does not adjust the minimum sleep time to compensate for the execution time of interrupt OB. However, any forced sleep interval (generated by the execution monitor) is subtracted from the sleep interval generated by the minimum sleep time.
- The controller supports SFC47 ("WAIT"), which inserts a specified sleep interval for the priority class of the OB that calls SFC47. This sleep interval the OBs at the same or lower priority class as the OB that calls SFC47, but an OB in a higher priority class can interrupt this sleep interval. You

can use SFC47 to create sleep time that can be interrupted so that the controller can avoid jitter when handling any interrupts that are critical for the application.



Tuning Strategy

As you test the performance of the controller during the development phase of your project, consider the following strategy for adjusting the sleep time:

1. Set the minimum sleep time parameter to 0 and run the control program. This allows you to determine whether there is unacceptable jitter in the scan cycle.
2. To reduce any unacceptable jitter, first use the tuning panel to increase the minimum sleep time and observe the effect on cycle time and CPU usage.
3. If the amount of jitter is still unacceptable, review the sections of the control program that are being affected by the jitter. If possible, have your control program call SFC47 to add sleep time.
4. To further reduce any jitter, increase the execution time limit to the maximum possible execution time for your control program.

If the sleep management techniques do not provide adequate improvement in reducing jitter, consider increasing the priority for the controller. (The priority of the controller is not the same as the priority class of an OB.)

Sample Interaction of the Execution Monitor and the Minimum Sleep Time

To help explain the tools for managing the sleep time of the controller, the following example shows how the execution monitor and the minimum sleep time can interact:

- The first sample shows the sleep time that would be generated by the execution monitor alone, with no minimum sleep time added to the free cycle.
- The second sample shows how the execution of the free cycle is affected by adding a minimum sleep time to the scan cycle.

The following example describes the execution of a control program that uses OB1 to start a 1-second timer, and then check the timer after an elapsed time of 1 second (1000 ms). The controller has been configured with the following parameters:

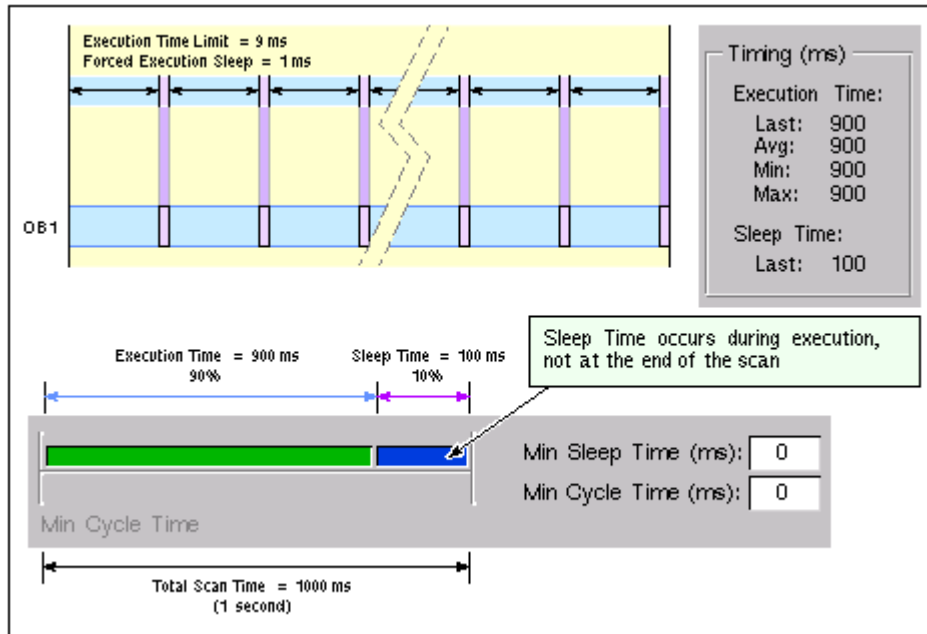
Parameter	Value
Execution Time	OB1 takes 900 ms to execute.
Minimum Sleep Time	0 ms
Minimum Cycle Time	0 ms
Maximum Execution Load	90% (uses the default wake/sleep algorithm)
Execution Time Limit	9 ms (uses the default value)
Forced Execution Sleep	1 ms (uses the default value)

Sleep Time Generated by the Execution Monitor (Minimum Scan Time = 0)

If you set the minimum sleep time parameter to 0, the controller uses the execution monitor alone to provide sleep time. The figure shows the operation of the execution monitor, using the default values.

The execution monitor suspends the execution of OB1 for 1 ms after every 9 ms of execution by default in order to enforce a limit of 90% execution load (CPU usage). For every 1 second of elapsed clock time, the default execution time for OB1 is 900 ms, with forced sleep intervals totaling 100 ms.

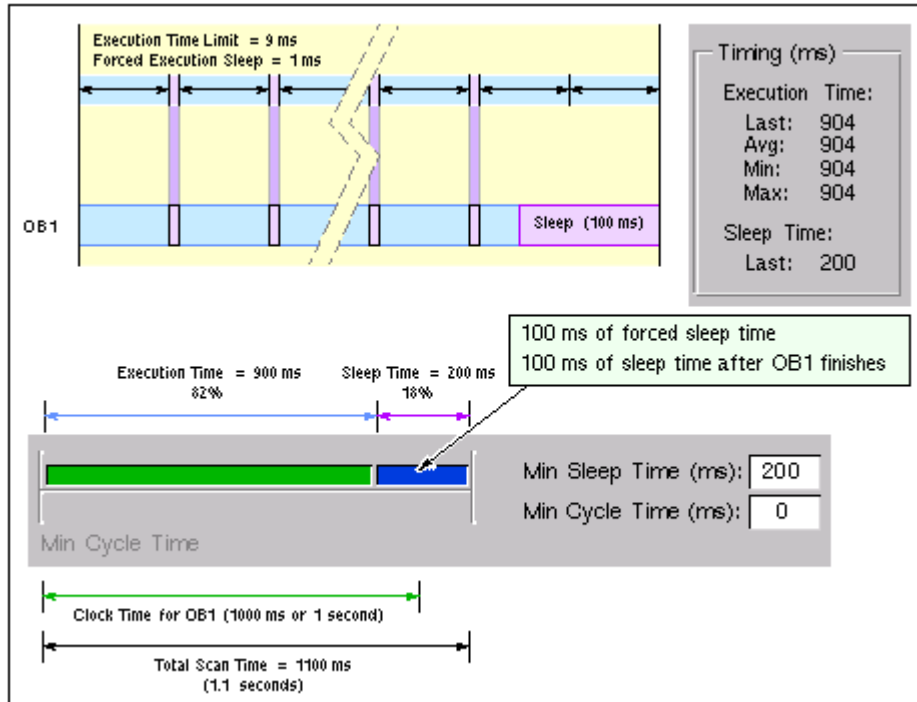
Notice that the sleep time occurs at intervals within the execution of OB1.



Adding a Minimum Sleep Time for the Free Cycle

This figure shows how changing the minimum sleep time from 0 to 200 affects the execution of OB1. The execution monitor still forces 100 ms of sleep time to occur during the execution of OB1. With the minimum scan time parameter set to 200 ms, the controller then sleeps for only another 100 ms, for a combined total of 200 ms, before starting the next free cycle.

The total scan time increases to approximately 1100 ms: the execution time (900 ms) for OB1, the forced sleep time (100 ms), and the sleep time at the end of the scan cycle (100 ms).



Adjusting the Minimum Sleep Time and Cycle Time

The tuning panel provides the following parameters that allow you to manage the sleep time of the free cycle (priority class 1, or OB1):

- Minimum Cycle Time (in milliseconds) sets the minimum number of milliseconds from the start of one free cycle to the start of the next free cycle. This value must be greater than the execution time before it causes any sleep time to occur within the free cycle. You use STEP 7 to configure the minimum cycle time for the controller when you create the system (hardware) configuration. You can use the tuning panel to adjust the minimum cycle time, but any changes are discarded when you shut down the controller. However, you must use STEP 7 to make the changes permanent.
- Minimum Sleep Time (in milliseconds) determines how much sleep time is available during the free cycle (OB1) for allowing higher priority OBs and other applications to use the resources of the computer. The controller automatically saves any changes to the minimum sleep time made with the tuning panel. You do not use STEP 7 to make any change to the minimum sleep time permanent.

Contents of this topic:

- Parameters That Affect the Sleep Time for the Free Cycle
- Hints

The execution of the free cycle is affected by both the minimum sleep time and the minimum cycle time values.

- The minimum cycle time by itself results in a fixed scan cycle time with a variable sleep time (if the minimum cycle time is large enough to accommodate the execution time plus the sleep time).
- The minimum sleep time by itself results in a fixed sleep time with a variable scan time, depending on the length of the execution time.

The minimum sleep time value guarantees that a configured amount of sleep time occurs within each free cycle, even if the value for the minimum cycle time is too small. The controller releases control of the CPU for a sleep interval. This sleep interval is the larger of either the configured minimum sleep time value or a sleep time that is computed from the minimum cycle time parameter.



Warning

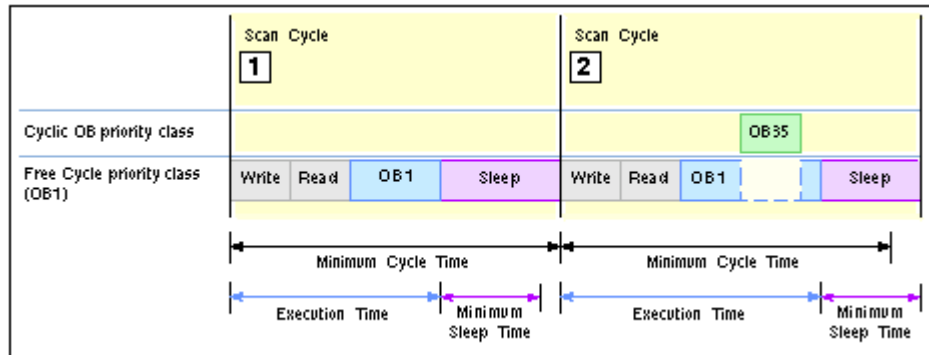
If you set the minimum scan time to a value larger than the watchdog time, WinLC goes to STOP mode during the first scan at the end of the watchdog time interval.

Causing the controller to go to STOP mode unexpectedly can cause process equipment to function erratically and possibly cause damage to equipment or injury to personnel.

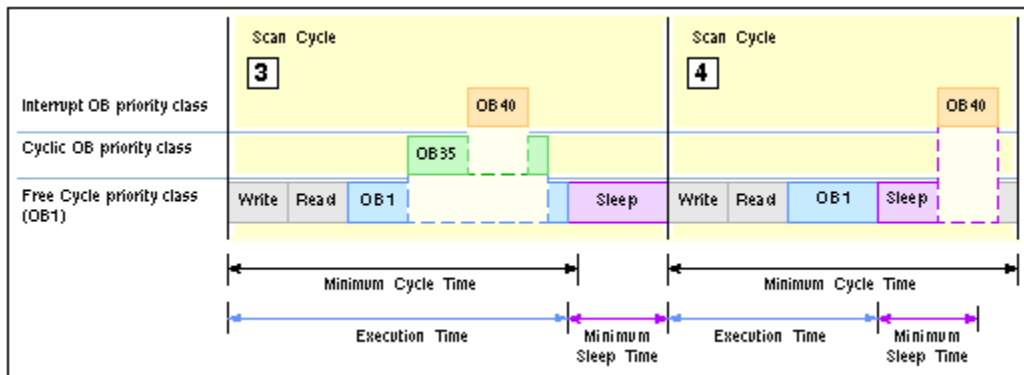
Do not set the minimum scan cycle time to be longer than the scan cycle monitoring time (the watchdog time) configured in the STEP 7 Hardware Configuration Editor.

Parameters That Affect the Sleep Time for the Free Cycle

The following figures explain the interaction between the execution time, the minimum sleep time, and the minimum cycle time parameters.



- 1 For the first sample scan shown in the example above, the execution time plus minimum sleep time is less than the minimum cycle time. In this case, the controller increases the sleep time until the minimum cycle time is achieved.
- 2 For the second sample scan shown in the example above, the execution of OB35 increases the execution time, and the execution time plus the minimum sleep time is greater than the minimum cycle time. In this case, the controller waits the minimum sleep time before starting the next scan.



- 3 For the third sample scan shown in the example above, the controller executes both a cyclic interrupt (OB35) and an I/O interrupt (OB40). The execution time exceeds the minimum cycle time, and the controller waits the minimum sleep time before executing the next scan.
- 4 For the fourth sample scan shown in the example above, the controller executes OB40 during the sleep time after OB1 has finished. In this case, the controller waits until the minimum cycle time before starting the next scan.

Because the execution of OB40 does not reset the minimum sleep time counter, it is possible that the controller does not provide sufficient sleep time to allow other Windows applications to be processed. You must then use other methods for ensuring that the controller provides a sufficient amount of sleep time.

Hints

You can use the following techniques to adjust controller performance using the minimum sleep time and minimum cycle time parameters:

- Use the tuning panel to test values for the minimum cycle time. After you have determined the optimum value for the minimum cycle time, use STEP 7 to update and download the system configuration for the controller.

Changing the operating mode from STOP to RUN deletes any value entered by the tuning panel and resets the minimum cycle time to the value stored in the system configuration.

- To ensure that the controller executes the scan cycle on a fixed schedule, use the minimum cycle time parameter.
- To ensure that there is always a sleep interval even if the execution time changes, set the minimum cycle time to 0 (the default value) and modify the minimum sleep time as needed. Modifying the minimum sleep time is especially useful during the development of your control program.

When you are tuning the operation of the controller, be aware that the following situations can increase the time required to complete the scan cycle:

- The controller executes other OBs (such as OB20 and OB35) with higher priorities than OB1.
- You use STEP 7 to monitor and debug the control program.
- You use a variable table (VAT) with STEP 7 to display the status of the control program.
- An application with a higher priority is running on your computer.
- The controller interacts with an HMI interface, such as WinCC.

Additional Methods for Managing the Sleep Time

- Using SFC47 to add sleep time in the control program
- Adjusting the sleep-monitoring algorithm of the execution monitor

Using SFC47 to Add Sleep Time in the Control Program

SFC47 (WAIT) inserts sleep time into the execution of the control program, allowing you to manage the sleep time for a control program by inserting the sleep time in a specific priority class. When you call SFC47 from your control program, the controller suspends the execution of the OB for a specified number of microseconds and sleeps. During this sleep period, the controller can interrupt this sleep period to execute an interrupt OB. Because an OB with a higher priority class can interrupt the sleep time, your control program to handle higher priority OBs with less chance of jitter.

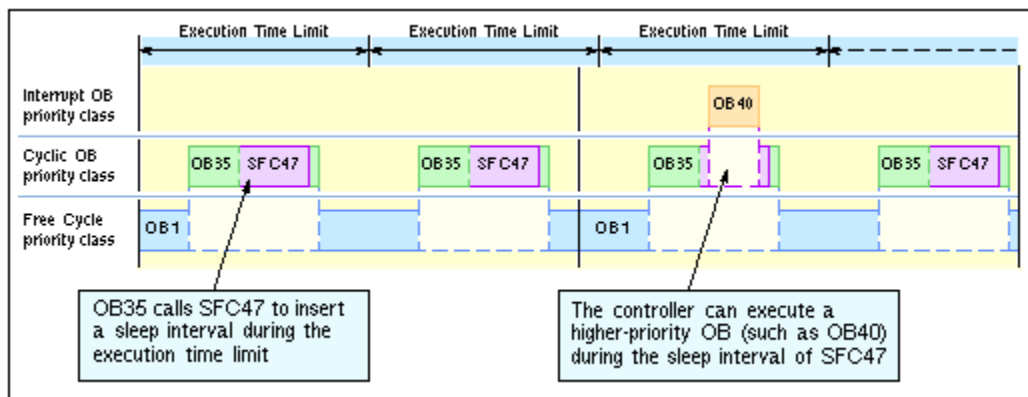
Typically, you call SFC47 from a cyclic OB (such as OB35) that starts within the execution time limit of the execution monitor.

For more information, refer to the example: Avoiding Jitter in the Start Time of an OB

To provide greater control over when the sleep time occurs, you can use SFC47 to insert sleep time into your control program. Calling SFC47 in the control program also allows you to define which OBs are affected by setting the priority class of the OB that calls SFC47.

As shown in the following figure, you can use SFC47 to insert a sleep interval that can satisfy the execution monitor and still allow the controller to handle an interrupt OB. By using a cyclic OB (such as OB35) to call SFC47, you can ensure that the sleep interval occurs within the execution time limit of the execution monitor.

The sleep time parameter is rounded up to the nearest millisecond.



Additional Methods for Managing the Sleep Time

- Adjusting the Minimum Sleep Time and Cycle Time
- Adjusting the Sleep-Monitoring Algorithm of the Execution Monitor

Adjusting the Sleep-Monitoring Algorithm of the Execution Monitor

The execution monitor uses a sleep-monitoring algorithm to ensure that the controller does not exceed a configurable **maximum execution load** for the CPU usage within a **monitor interval**.

The monitor interval is calculated as the amount of time such that the maximum load percentage of the monitor interval equals the entered **execution time limit**. The execution monitor calculates the **forced execution sleep time** as the difference between the monitor interval and the execution time limit.

The execution monitor determines whether to insert a forced execution sleep time if the OB execution exceeds the execution time limit.

If there is sufficient sleep time within the monitor interval, the execution monitor does not affect the execution of the program. Otherwise, the execution monitor forces a sleep interval. The default execution load is 90%, and the default execution time limit is 9 ms. For the default settings, the execution monitor calculates a monitor interval of 10 ms and a forced sleep interval of 1 ms.

The execution monitor runs asynchronous to the scan cycle and measures the amount of sleep time that occurs within the monitor interval and enforces a minimum sleep interval.

- If the scan cycle (execution time plus sleep time) is shorter than the monitor interval and the sleep time is greater than or equal to the forced sleep value: The execution monitor does not force a sleep interval.
- If the scan cycle is longer than the monitor interval: The execution monitor forces the controller to sleep for the required amount of time. Because the execution monitor runs in a higher priority class than any OB, the controller cannot interrupt the forced sleep interval. This could delay the start of an interrupt OB, such as OB35 or OB40.

Use the tuning panel to configure the parameters for the sleep-monitoring algorithm of the execution monitor.

For more information, see the example: [Avoiding Jitter in the Start Time of an OB](#)

Contents of this topic:

- Operation of the Execution Monitor
- Parameters of the Sleep-monitoring Algorithm
- Configuring the Parameters of the Sleep-Monitoring Algorithm
- Situations that Cause the Execution Monitor to Force a Sleep Interval
- Situations that Prevent the Execution Monitor from Providing Sufficient Sleep Time

In addition to the sleep time that is added to the scan cycle (based on the minimum sleep time and minimum cycle time parameters), the execution monitor uses a sleep-monitoring algorithm that is based on a maximum execution load (percentage of CPU usage). For the default execution load (90% CPU usage), the execution monitor measures the length of time that the controller sleeps during the monitor interval of 10 ms and ensures that the controller sleeps for at least 1 ms.

By measuring the sleep time, the execution monitor ensures that the controller allows the other applications to access the computer resources while the controller sleeps. The execution monitor also provides the safety net in cases where there are programming errors (for example, an infinite loop in OB100) that are not handled with other mechanisms.

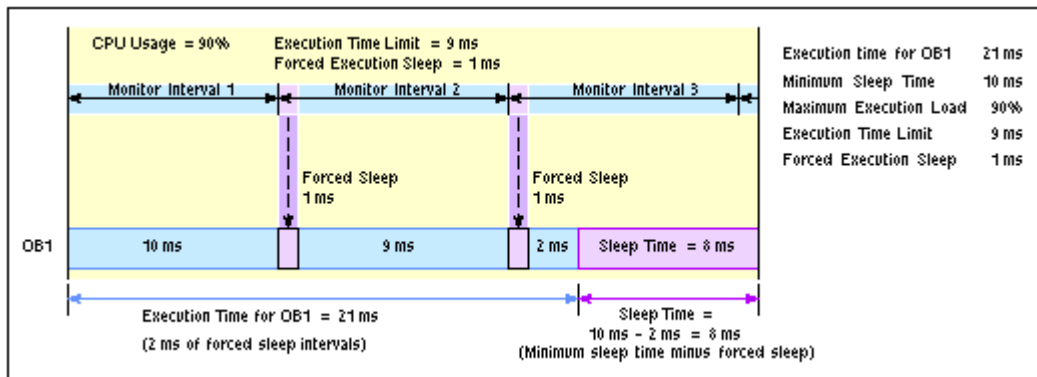
The difference between the forced sleep intervals and the minimum sleep time is that the controller can interrupt the minimum sleep time to handle interrupts (such as OB35 or OB40), but cannot interrupt the forced execution sleep time.

When the execution monitor forces a sleep interval, the following actions occur:

- The controller immediately suspends the execution of the OB for the forced sleep interval. By forcing a sleep interval, the execution monitor increases the actual time between starting and finishing the OB being executed.
- The controller cannot respond to the start event for any interrupt OB until the end of the forced sleep interval. Delaying the start of the OB (for example, OB35 or OB40) until the end of the forced sleep interval creates jitter or latency in the actual start time for the OB.

Operation of the Execution Monitor

The following figure shows how the execution monitor might affect a control program. Because the execution time for OB1 in this example is greater than the execution time limit, the execution monitor inserts a 1-ms sleep interval after the first two monitor intervals. However, the execution monitor does not insert a forced sleep interval in the third monitor interval because the controller sleeps longer than the required forced sleep interval as required by the configured minimum sleep time.



Note

The execution monitor runs asynchronous to the scan cycle. The example above shows the execution monitor measuring time from the beginning of the scan cycle, but because the execution monitor runs asynchronous to the control program, the beginning of the execution time limit of the execution monitor does not necessarily coincide with the beginning of the scan cycle.

Parameters of the Sleep-Monitoring Algorithm

The sleep-monitoring algorithm of the execution monitor uses the following parameters:

Parameter	Description
Execution Time Limit	<p>This value defines the maximum time (in microseconds) that the execution monitor allows for OB execution before exceeding the configured maximum execution load (CPU usage) of the monitor interval.</p> <p>To determine the CPU load caused by the execution of the control program, the execution monitor measures the time that the controller sleeps during the monitor interval. If the controller does not use a sufficient amount of sleep time (indicating that the CPU load exceeds the maximum execution load), the execution monitor forces the controller to sleep for the remainder of the required forced execution sleep time.</p> <p>The default value is 9000 microseconds (9 ms).</p> <p>Note: If you set this value greater than approximately 50000 (50 ms), you may observe jitter in Windows applications and in response to the mouse or keyboard. Test that the execution time limit you choose is appropriate for your application.</p>
Maximum Execution Load	<p>This value defines the maximum percentage of CPU usage that is allowed for the controller to execute OBs during each monitor interval.</p> <p>The default value is 90%.</p>
Forced Execution Sleep	<p>This read-only field shows how much sleep time (in microseconds) the execution monitor requires during the monitor interval to satisfy the requirement for the maximum execution load. The execution monitor subtracts any controller sleep time that occurs during a monitor interval from the forced execution sleep time to determine how much sleep time (if any) to force.</p> <p>The forced execution sleep time is a calculated number based on the execution time limit and the maximum execution load. The execution monitor corrects this value as required, depending on the capability of the operating system configuration to have timers operate at the specified intervals.</p> <p>The default value is 1000 microseconds (or 1 ms).</p>

The execution monitor uses the execution time limit and the maximum execution load to calculate the forced execution sleep. For example, the execution monitor uses the 90% usage rate and the 9-ms execution time limit to calculate a 1-ms sleep interval. In this case, the monitor interval is 10 ms such that 90% of the monitor interval corresponds to the entered execution time limit (9 ms).

During the monitor interval, the execution monitor measures the actual amount of time that no OBs are executing (the sleep time).

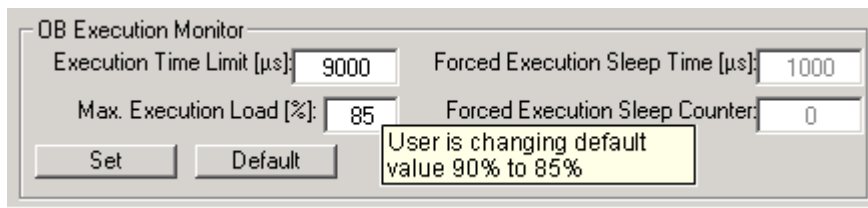
- If the controller sleeps **longer** than the sleep interval (forced execution sleep time), then the execution monitor restarts another monitor interval and does not affect the control program.
- If the controller sleeps **less** than the sleep interval (forced execution sleep time), then the execution monitor blocks the execution of any OBs for the **remainder** of the sleep interval.

Any control program sleep time imposed because of the sleep-monitoring algorithm is subtracted from the sleep time configured for the end of the free cycle as defined by the minimum sleep time parameter.

The default value for the “Execution Time Limit” interval is 9000 microseconds (or 9 milliseconds) and the default value for the “Forced Execution Sleep” interval is 1000 microseconds (or 1 millisecond). This ratio ensures that the control program execution cannot use more than 90% of the CPU time in any of the worst case situations described above.

Configuring the Parameters of the Sleep-Monitoring Algorithm

The parameters of the sleep-monitoring algorithm of the execution monitor are configurable from the tuning panel:



To change the sleep-monitoring parameters, follow these steps:

1. Enter values in the Execution Time Limit and the Max. Execution Load fields. You can change one of the fields or both.
2. Click Set to set the parameters.

To restore the default sleep-monitoring parameters, follow these steps:

1. Click Default to display the default parameters.
2. Click Set to set the default parameters.

Changes to the sleep-monitoring parameter take effect when the controller is in RUN/RUN-P mode.

Situations that Cause the Execution Monitor to Force a Sleep Interval

The controller must relinquish control of the CPU long enough to satisfy the maximum execution load. Typically, the sleep time that is added to the end of the scan cycle allows sufficient time for the operating system to process the other Windows applications. However, some situations may require that the execution monitor force a sleep interval.

Condition	Description
Execution time for the control program exceeds the execution time limit	The configured minimum sleep time for the free cycle occurs after OB1 finishes. If the execution time is longer than the execution time limit, the execution monitor forces a sleep interval because the controller did not sleep for the required amount within the monitor interval.
Minimum sleep time is insufficient for the maximum execution load	Even when the scan cycle is less than the execution time, the minimum sleep time may not provide enough sleep time. In this case, the controller would exceed the maximum execution load. The execution monitor forces an additional sleep interval to ensure that the operating system can run the other applications.
Interrupt OBs reduce the sleep time	<p>To process an interrupt OB (such as OB35, OB40, or OB85), the controller can interrupt the sleep time for the scan cycle. This reduces the time that the controller actually sleeps and can cause the controller to exceed the maximum execution load, which affects the performance of the other Windows applications.</p> <p>By forcing a sleep interval, the execution monitor ensures that the other Windows application can be processed.</p>

Situations that Prevent the Execution Monitor from Providing Sufficient Sleep Time

In some cases, a high execution time limit can prevent the execution monitor from managing the sleep time of the control program adequately. Under the following conditions, the control program utilizes too much CPU time, which can result in jitter in Windows response time to the mouse, keyboard, or other applications. For either case, the problem can be resolved by lowering the execution time limit.

Condition	Description
Execution time for the startup OB (OB100 or OB102) and the configured execution time limit exceed ~50 ms	<p>During startup, the controller turns the watchdog timer off and cannot handle a program error, such as a loop in the logic of the OB or an excessively long initialization routine.</p> <p>Because the scan cycle does not provide any sleep time for the startup OB (such as OB100), the execution monitor cannot relinquish CPU time for other applications. If the startup OB executes for more than ~50 ms, jitter can occur in Windows response time to the mouse, keyboard, or other applications.</p>
Execution time for the control program and the configured execution time limit exceed ~50 ms	<p>Whenever the operating system has to wait more than ~50 ms to process the other Windows applications, the performance of those applications can be noticeably affected. This can be a problem for an OB1 with a long execution time, especially if other OBs (such as OB35 or OB40) extend the execution of OB1.</p> <p>Because the sleep time is added at the end of the scan cycle, and the execution time limit is set to a high value, the sleep intervals are then spaced too far apart for the other Windows applications to perform naturally.</p>

Additional Methods for Managing Sleep Time

- Adjusting the minimum sleep time and minimum cycle time parameters
- Inserting sleep time into the control program (SFC47 "WAIT")

Example: Avoiding Jitter in the Start Time of an OB

The following example discusses two possible solutions for a program that experiences jitter in the start of a cyclic interrupt (OB32 to OB36).

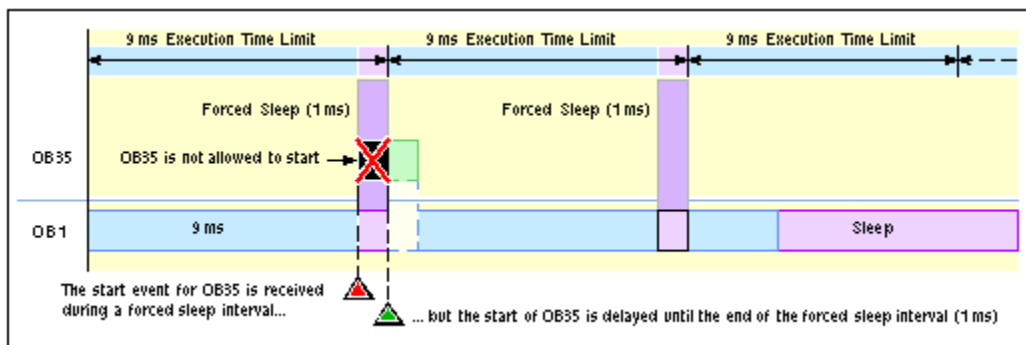
- Inserting a sleep interval into the execution of your control program. For this solution, you call SFC47 (“WAIT”) and specify the length of time to sleep. The controller can interrupt this sleep interval to process other OBs.
- Changing the sleep-monitoring algorithm of the execution monitor. For this solution, you use the tuning panel to change the execution time limit.

Scenario

The following example describes the execution of a control program that consists of OB1 and OB35. OB1 takes 20 ms to execute, and OB35 starts every 100 ms and takes 1 ms to execute. The controller has been configured with the following parameters:

Parameter	Value
Execution Time for the Control Program	OB1: 20 ms, and OB35: 1 ms
Minimum Sleep Time	10 ms (uses the default value)
Minimum Cycle Time	0 ms (uses the default value)
Maximum Execution Load	90% (uses the default wake/sleep algorithm)
Execution Time Limit	9 ms (uses the default value)
Forced Execution Sleep	1 ms (uses the default value)

The sleep time (10 ms) is added to the scan cycle after OB1 has finished. However, because the execution time for OB1 (20 ms) exceeds the execution time limit (9 ms), the controller exceeds the configured maximum execution load (90%) by not sleeping during the execution time limit. Therefore, the sleep-monitoring algorithm forces the controller to sleep for 1 ms after every 9 ms that OB1 executes. As shown in the following figure, this forced sleep can cause a variance or jitter of up to 1 ms between time that the start event and the time that the controller starts to execute OB35. This jitter happens because all controller operations are suspended during a forced sleep interval. Similarly, OB35 could be suspended for 1 millisecond if the end of the execution time limit interval occurs while OB35 is executing.



For many applications, a 1-ms jitter might be acceptable. However, you have several options for removing this jitter:

- You can modify the control program to call SFC47 and insert sleep time that can be interrupted by OB35.
- You can adjust the parameters for the sleep-monitoring algorithm to avoid the jitter caused by the execution monitor.

Solution 1: Insert a sleep interval into the execution of your control program

You could avoid the forced sleep interval by using SFC47 to add a periodic sleep interval that occurs within the execution time limit (for this example, 9 ms). This sleep interval not only ensures that the sleep-monitoring algorithm does not force the controller to sleep, but also allows the controller to suspend this sleep interval and execute any OB that has a higher priority than the OB that called SFC47.

For this example, you can use SFC47 to remove the jitter in OB35:

- By ensuring that SFC47 executes at a specified time. The control program calls SFC47 from an OB (such as OB36) that has a priority greater than OB1.
- By ensuring that OB35 executes as scheduled. You configure OB36 to have a lower priority than OB35.
- By ensuring a sufficient sleep interval during the execution time limit. You configure SFC47 to wait for 3 ms, which ensures a sleep interval of at least 2 ms.

To maintain a 50% ratio for CPU usage (20 ms execution time for OB1 with a 10 ms minimum sleep time), configure OB36 to run every 6 ms (so that OB1 executes for 6 ms, then sleeps for 3 ms). You can then change the minimum sleep time to 0 ms, unless you want to decrease the ratio for CPU usage.

To create an OB36 that calls SFC47 to create a 3 ms sleep interval:

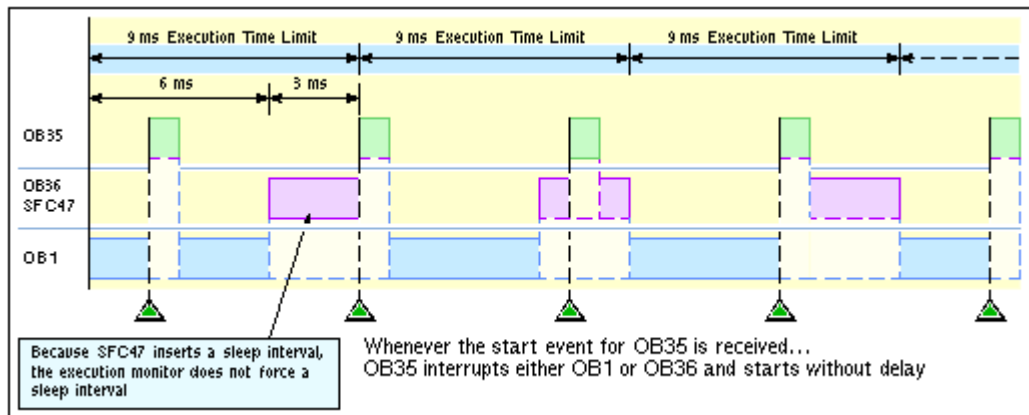
1. Using the STEP 7 Program Editor: Create an OB36 for your control program, and enter the following program:

```
CALL "WAIT" // SFC47 wait function
WT: 3000 // 3000 microseconds or 3 milliseconds
```

2. Using the STEP 7 Hardware Configuration tool, configure the priority level and execution time for OB36:

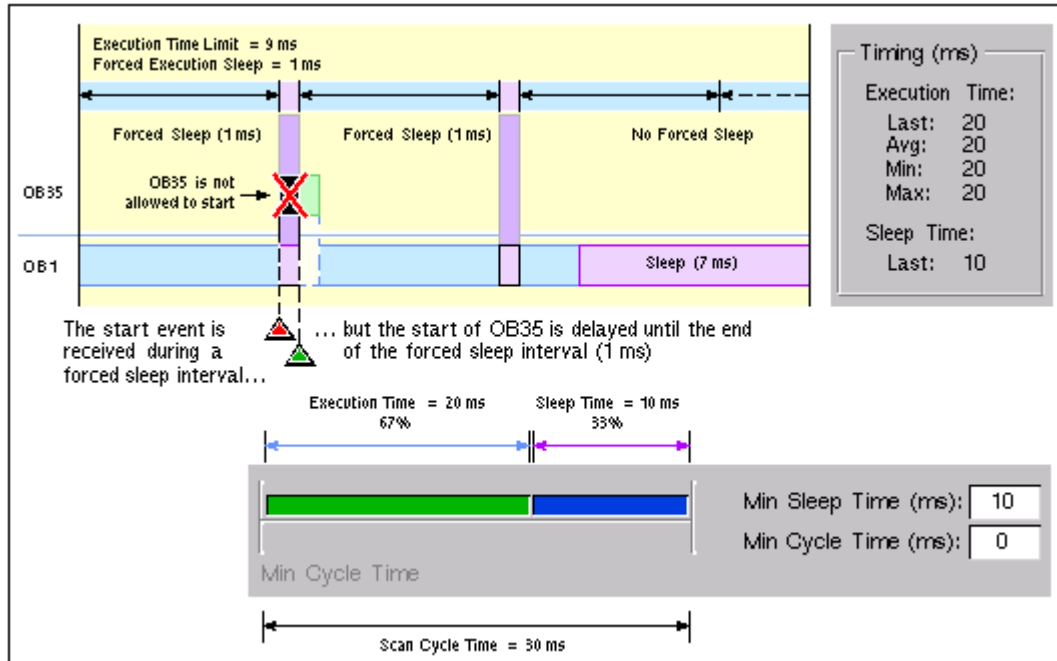
- Open the WinLC Properties dialog box and select the Cyclic Interrupt tab.
- Set the priority for OB36 to 2 (or any other priority lower than the priority for OB35).
- Configure OB36 to execute every 6 ms (by entering 6 in the Execution field).

The following figure shows how SFC47 affects the execution of the control program. Because OB36 ensures that the controller sleeps at least 1 ms within the 90% wake interval, the execution monitor does not insert a forced sleep interval. Therefore, OB35 executes without any delay or jitter.



Solution 2: Change the sleep-monitoring algorithm to eliminate the forced sleep interval

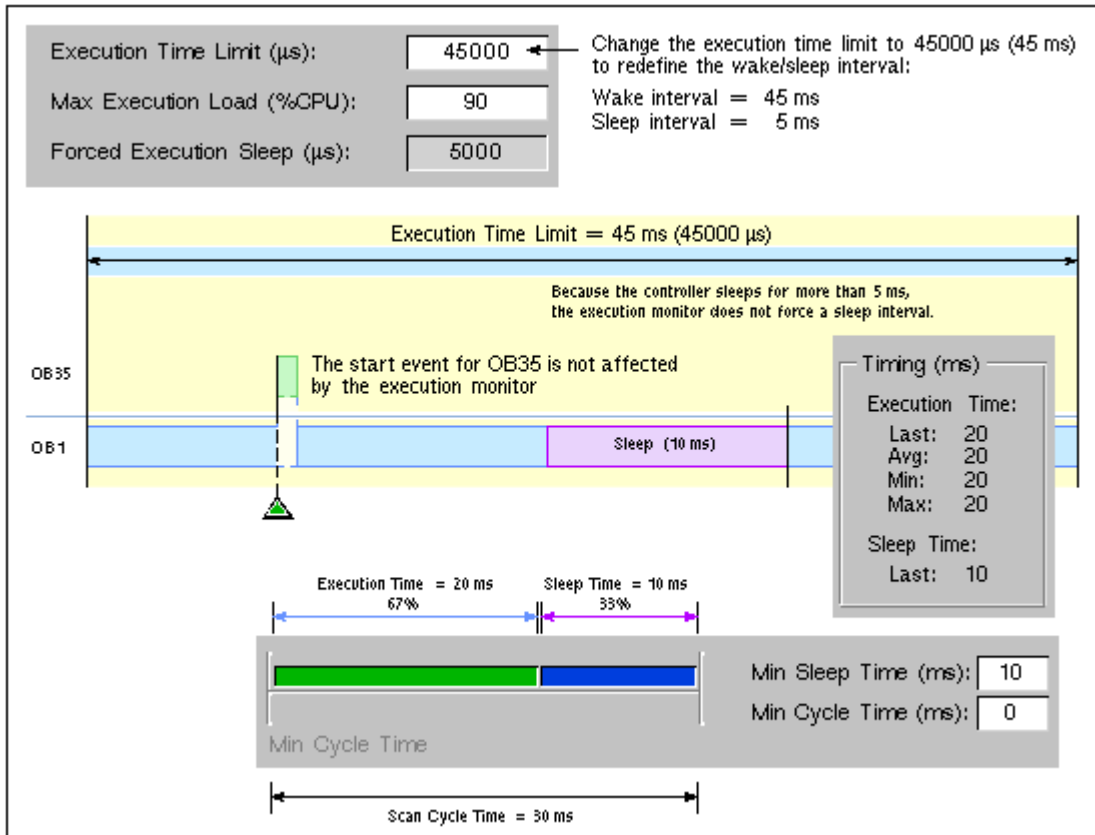
The following figure shows the jitter in the start time of OB35 and also shows the values displayed by the tuning panel. Notice that the tuning panel shows only the information about OB1. The tuning panel does not display information about OB35. For this example, the execution time for OB1 is 20 ms. With the minimum sleep time of 10 ms, the total free cycle time is 30 ms. OB35 and other interrupt OBs can make the total scan time more than this, depending on how fast the interrupt OBs execute.



By changing the parameters of the sleep-monitoring algorithm, you can configure the execution monitor to use the minimum sleep time in the free cycle. For example: if the longest total scan time for this example is less than 45 ms, change the execution time limit to 45000 microseconds (45 ms):

1. Open the tuning panel.
2. Change the execution time limit to 45000 (microseconds). For this example, do not change the value for the maximum execution load.
3. Apply the new value.

The following figure shows the effect of the changed execution time limit.




Advanced Topics






Connecting WinLC to the SIMATIC NET OPC Server

WinLC can use the SIMATIC NET OPC server to read and write data over the network. You use the following tools to configure the OPC connection for WinLC:

- OPC Scout for configuring the connection to the SIMATIC NET OPC server
- STEP 7 (HW Config and NetPro) for configuring the WinLC controller
- Station Configuration Editor for configuring the PC station

 The critical step most frequently overlooked is configuring the S7 connection for the OPC server in NetPro. After adding the connection for the OPC server, you must set the connection type to "S7 connection" and enter a Local ID for the connection.

Task Overview

	Step 1: Station Configuration Editor (SIMATIC NET) Add the OPC server to the PC station.
	Step 2: HW-Config (STEP 7) Add the OPC server to the configuration of WinLC.
	Step 3: NetPro (STEP 7) Add an S7 connection for the OPC server to the configuration of WinLC.
	Step 4: SIMATIC Manager (STEP 7) Download the configuration to the WinLC controller.
	Step 5: OPC Scout (SIMATIC NET) Connect WinLC to the OPC server.

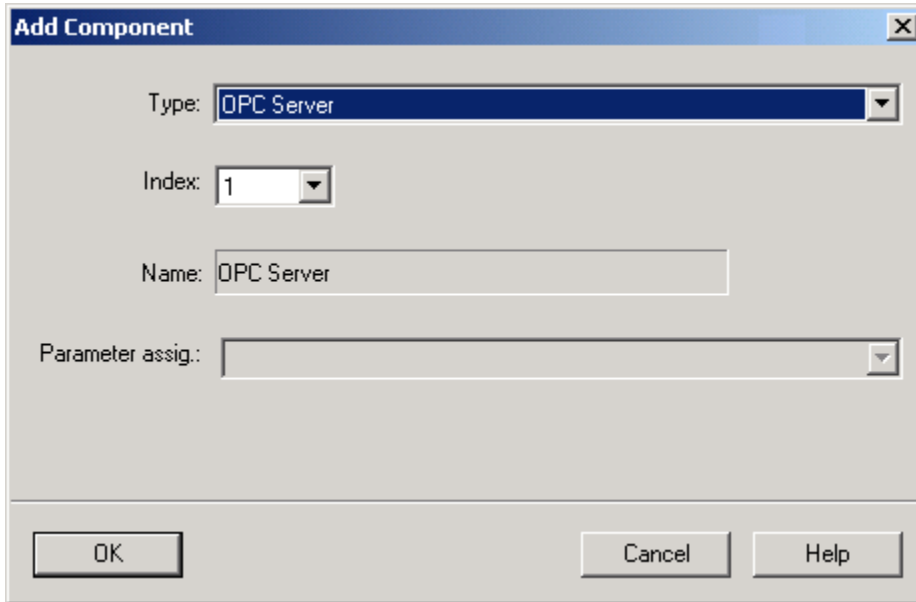
Step 1: Add the OPC Server to the PC Station

Station Configuration Editor (SIMATIC NET)

- Configure the OPC server for an index of the PC station.

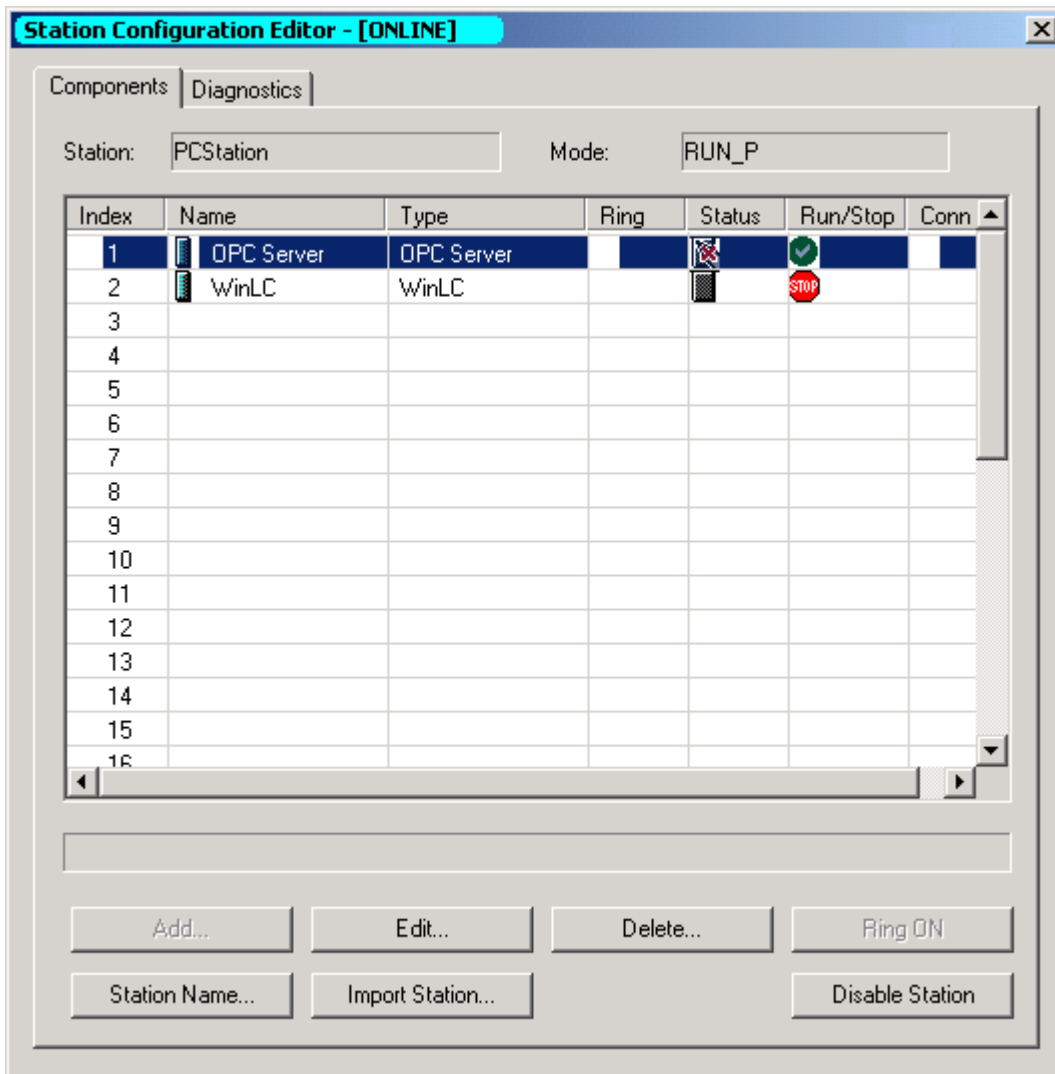
1. Open the Station Configuration Editor and select any index in the Station Configuration Editor.
2. Right-click the mouse to display the Add button. Click the Add button to display the Add Component dialog.
3. Select the following component type from the drop-down list:

OPC Server



The screenshot shows the 'Add Component' dialog box. The 'Type' dropdown is set to 'OPC Server'. The 'Index' dropdown is set to '1'. The 'Name' text box contains 'OPC Server'. The 'Parameter assign.' dropdown is empty. The 'OK', 'Cancel', and 'Help' buttons are visible at the bottom.

- Click OK to add the OPC server to the station configuration. The Station Configuration Editor displays the OPC Server in the index selected. (For this example, the OPC server is configured for Index 1.)
- Click OK to save the PC station configuration and to close the Station Configuration Editor dialog.

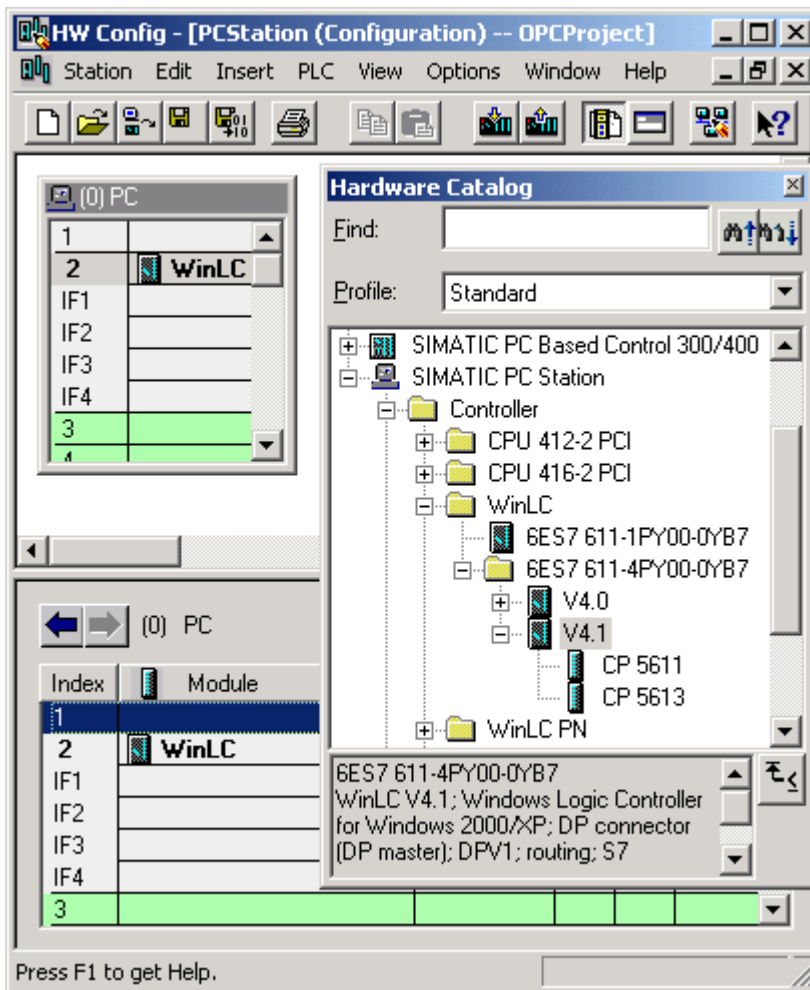


Step 2: Configure WinLC for Using the OPC Server

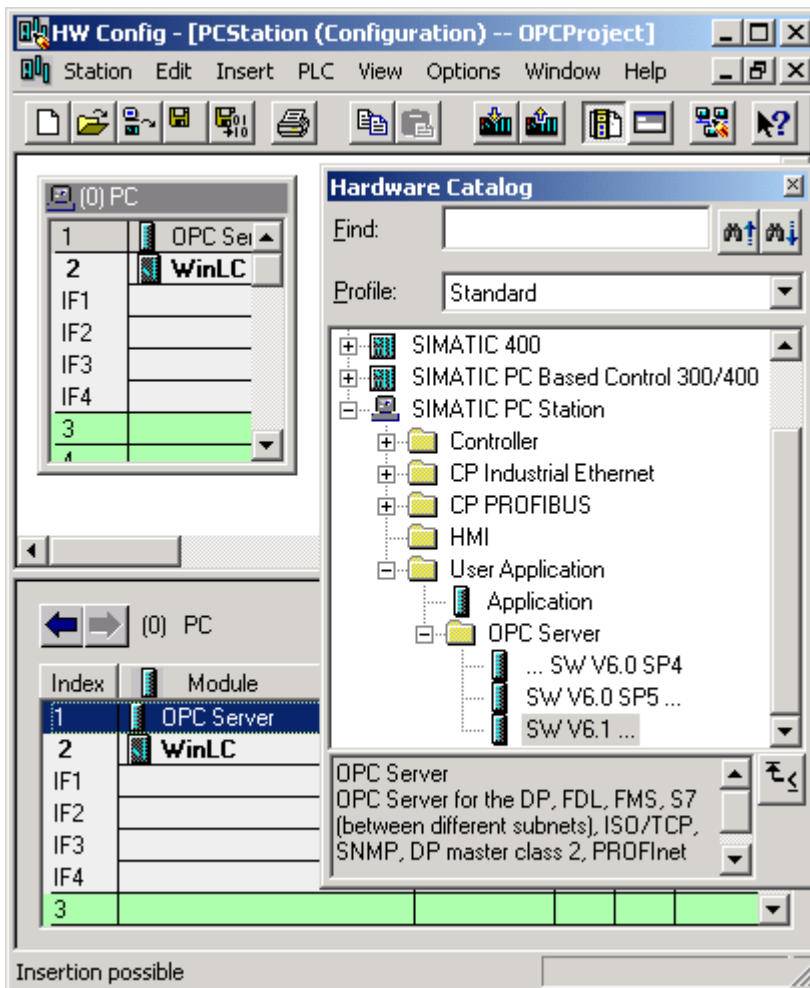
HW Config (STEP 7)

- Create a STEP 7 project for a PC station with WinLC.
- Insert the OPC server into the hardware configuration.
- Configure the OPC server.

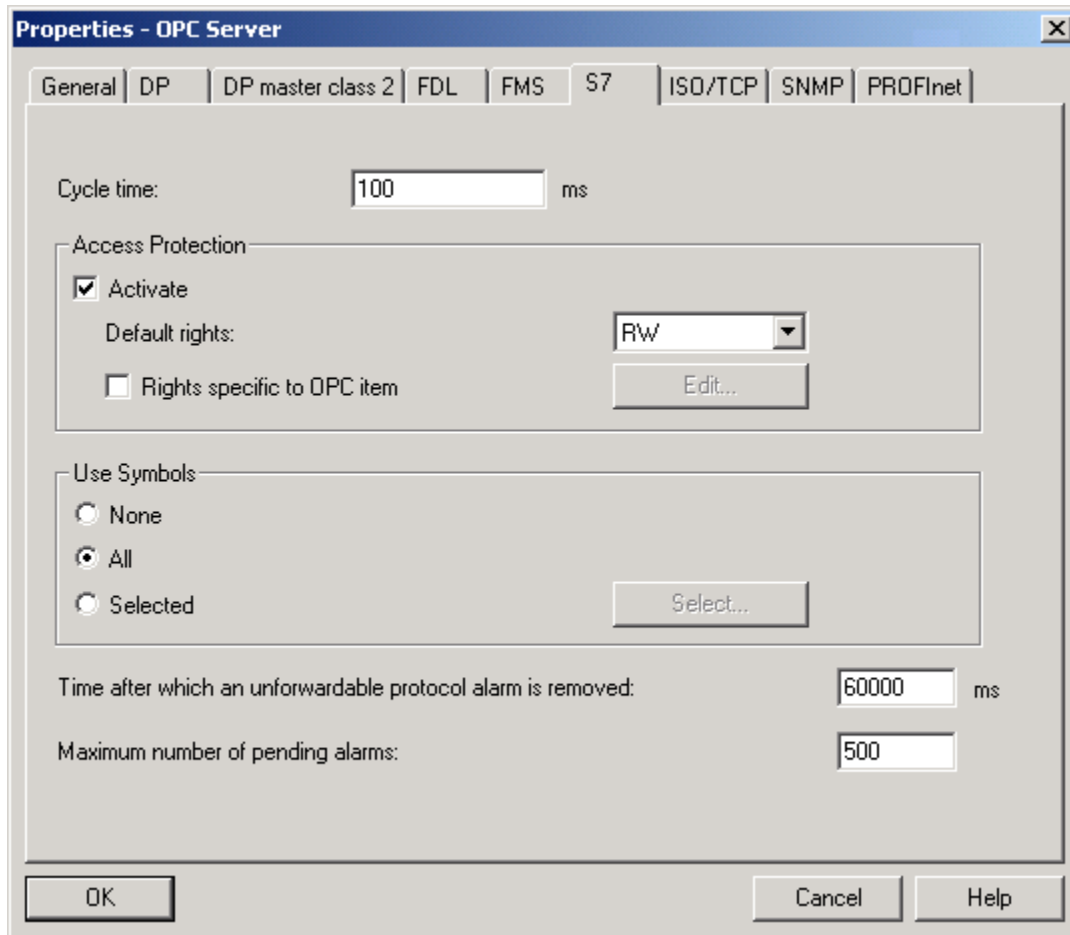
1. Open STEP 7 and create a project (for example, OPCProject).
2. Insert a SIMATIC PC Station with the same name as entered in the Station Configuration Editor. Open HW Config by double-clicking the Configuration icon for the PC Station.
3. Insert the WinLC controller in the same index as configured in the Station Configuration Editor. (For this example, a WinLC Basis controller is configured for Index 2.)



- Expand the User Application folder in the catalog.
- Expand the OPC Server folder and select the following component:
SW V6.1
- Drag and drop the SW V6.1 component to in the same index as configured in the Station Configuration Editor. (For this example, the OPC server is configured for Index 1.)
- Double-click the OPC Server entry (Index 1) to open the Properties dialog.



8. Click the S7 tab and select the Activate option (under Access Protection).
9. To use the STEP 7 symbols when configuring the connections in OPC Scout, select the option for All (or for Selected, to specify specific entries in the symbol table) under the Use Symbols field.
10. Click OK to close the Properties dialog.
11. Click the Save and Compile icon to create the hardware configuration for the PC station.

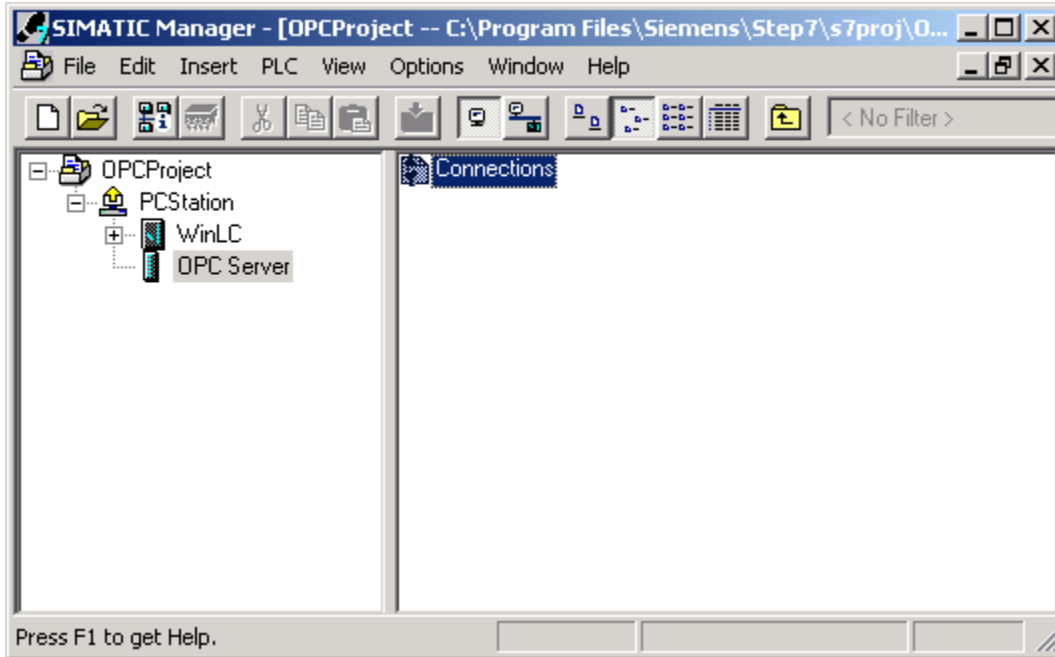


After you have compiled the configuration into the STEP 7 project, you can close HW Config and return to SIMATIC Manager.

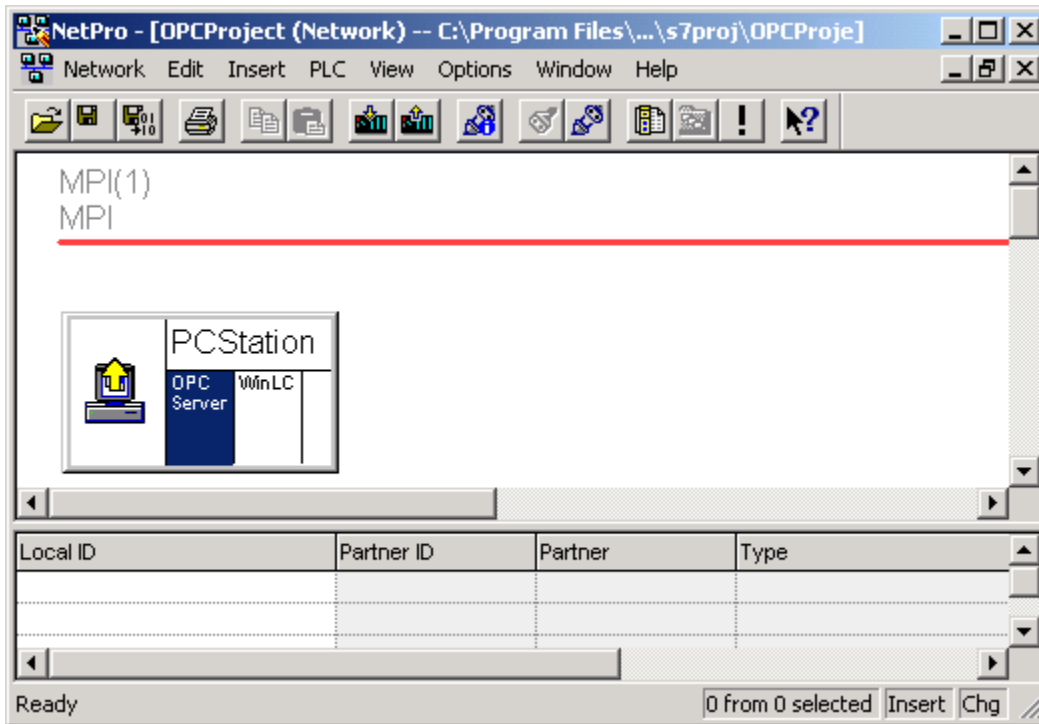
Step 3: Add the OPC Connection to the WinLC Configuration**NetPro (STEP 7)**

- Add a connection for the OPC server to the WinLC configuration.
- Configure the OPC server connection as an S7 connection.
- Assign a Local ID for the OPC server connection.

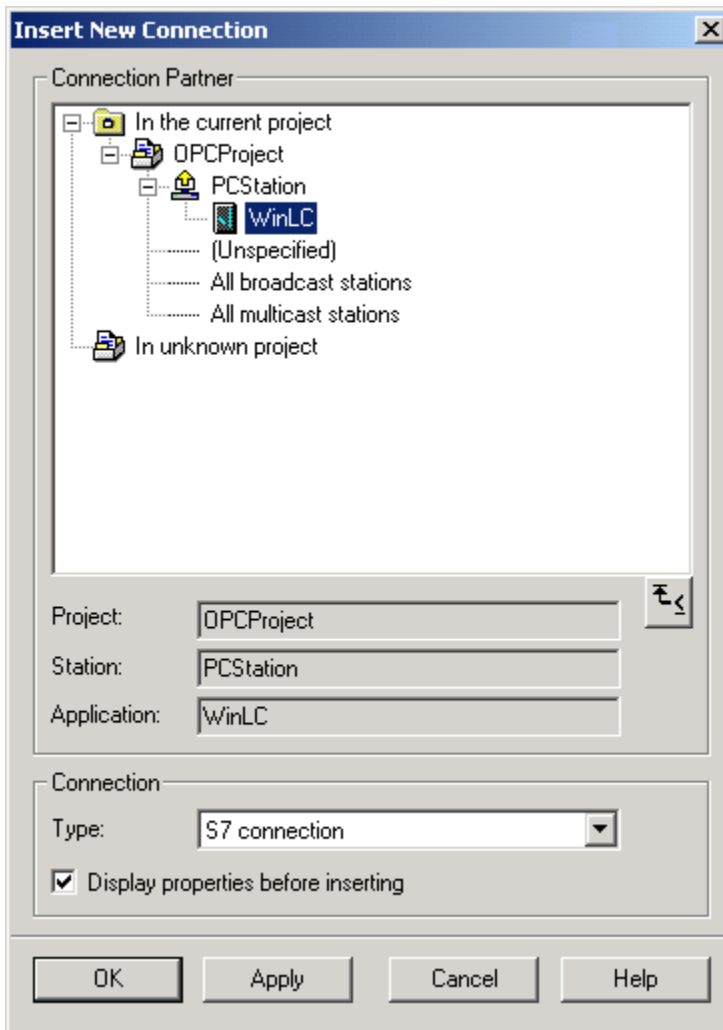
1. In SIMATIC Manager, browse to the OPC server and double-click the Connections icon to open NetPro.



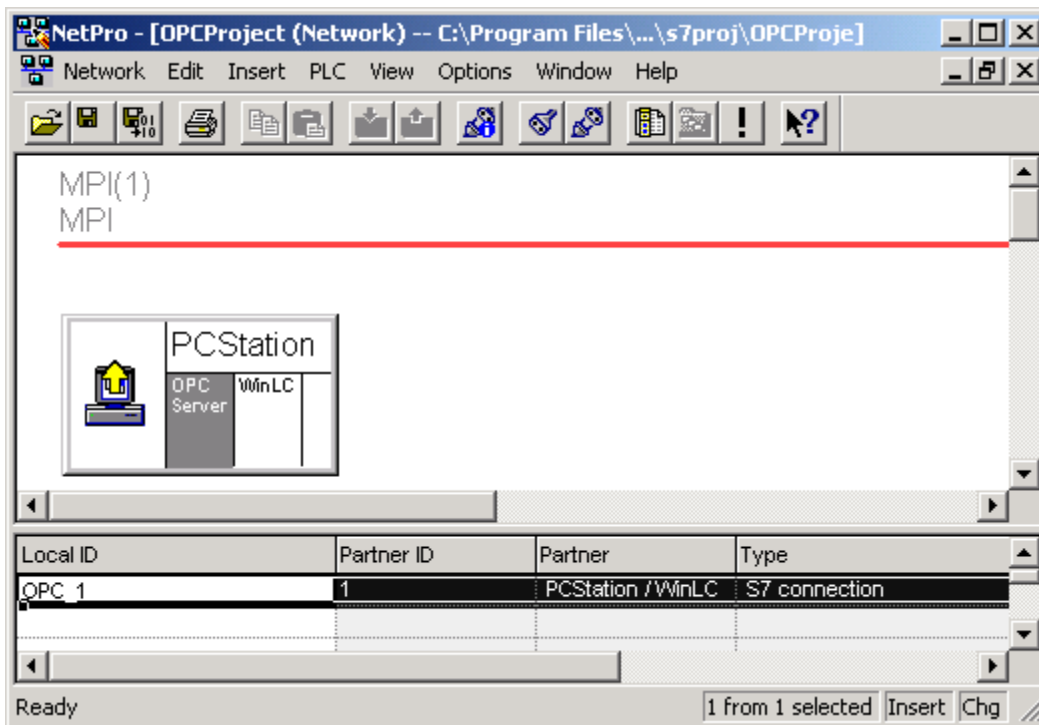
2. Select the OPC Server in the PC station.



3. Right-click the OPC server to display the context menu. Select the **Insert New Connection** menu command to open the Insert New Connection dialog.



4. Set the connection type to S7 connection and click OK to add the S7 connection for the OPC server. The Properties dialog for the S7 connection opens automatically.
5. Enter the Local ID for the S7 connection (such as OPC_1).
6. Click OK to add the S7 connection to NetPro.
7. Click the Save and Compile icon to save and compile your changes into the STEP 7 project.



After you have compiled the S7 connection for the OPC server into the STEP 7 project, you can close NetPro and return to SIMATIC Manager.

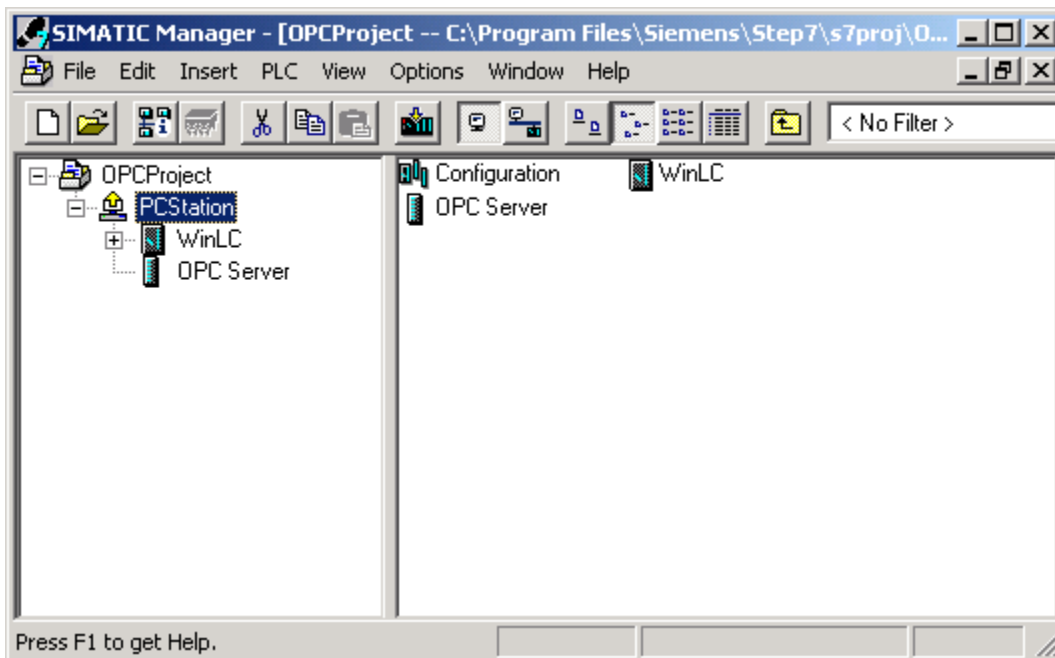
Step 4: Download the configuration to the controller** SIMATIC Manager (STEP 7)**

- Start the WinLC controller.
- Download the configuration.

To download the configuration, the WinLC controller must be running. To start the controller, select the **Start > SIMATIC > PC Based Control > WinLC** menu command or double-click the WinLC icon on the desktop.

After you have started the WinLC controller, you can download the configuration:

1. In SIMATIC Manager, select the SIMATIC PC Station icon.
2. Select the **PLC > Download** menu command or click the Download icon on the toolbar.



Step 5: Connect WinLC to the OPC Server

OPC Scout

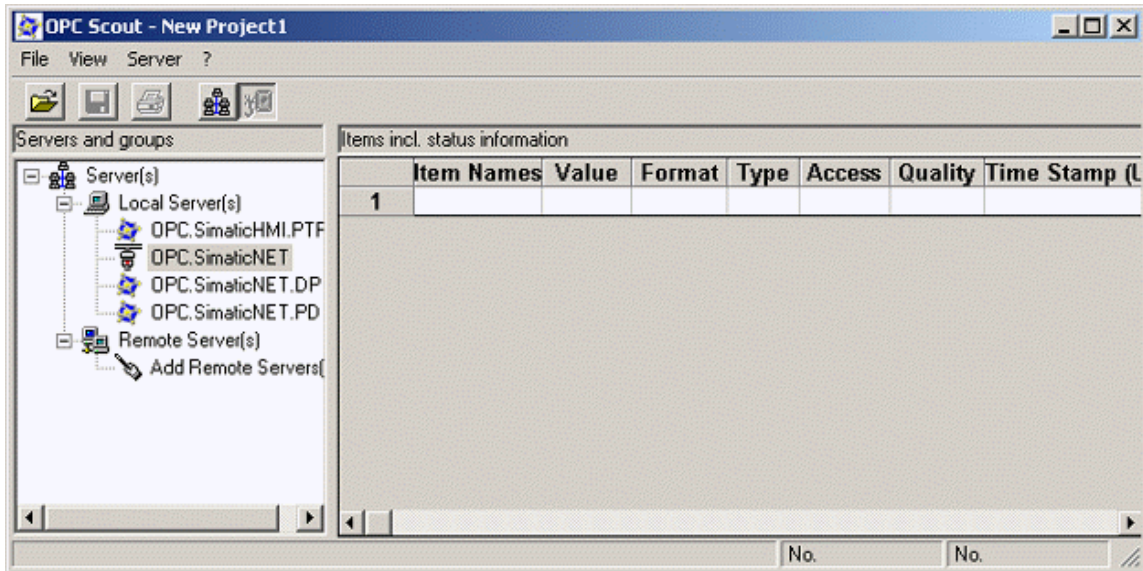
- Create an OPC Project.
- Add the connection to the SIMATIC NET OPC server.
- Define the items to be accessed through the OPC sever.

Opening an OPC Project

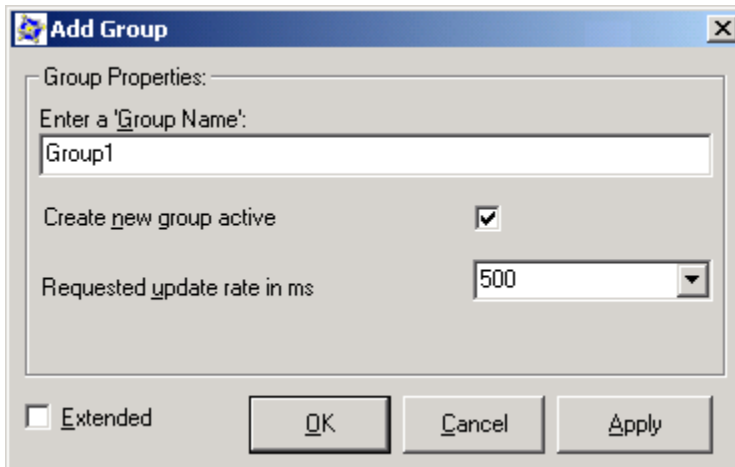
Select the **Start > SIMATIC > SIMATIC NET > PROFIBUS > SOFTNET PROFIBUS > OPC SCOUT** menu command to open a new project in OPC Scout.

Adding a Connection (Group) for the OPC Server

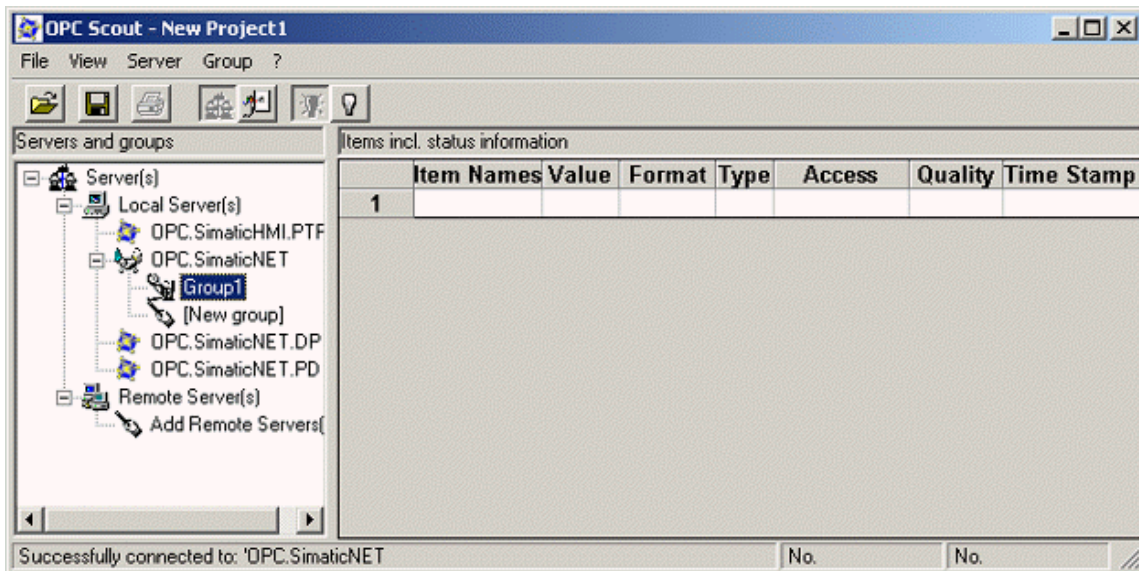
1. Expand the Local Server(s) directory in the Servers and Groups for the project.
2. Double-click the OPC.SimaticNet element to add a connection (or group) for the SIMATIC NET OPC server.



- In the Add Group dialog, enter the Group Name for the connection (for example, Group1).



- Click OK to add the group to the OPC server. OPC Scout adds the connection (Group1) to the OPC server.

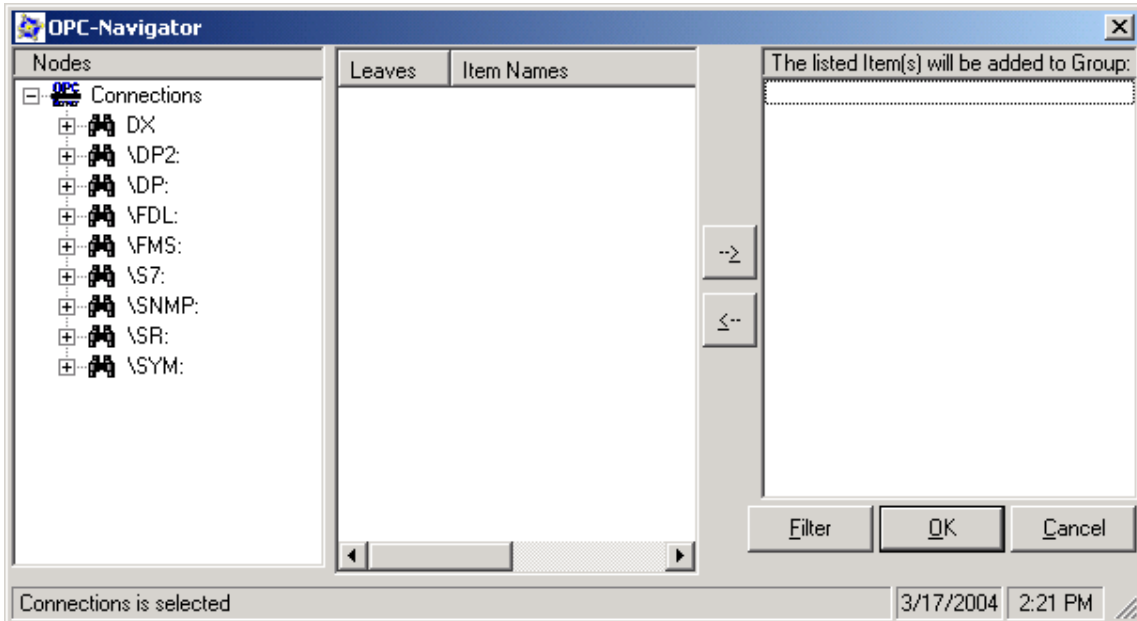


Configuring the Items to be Accessed (Using Absolute Addressing)

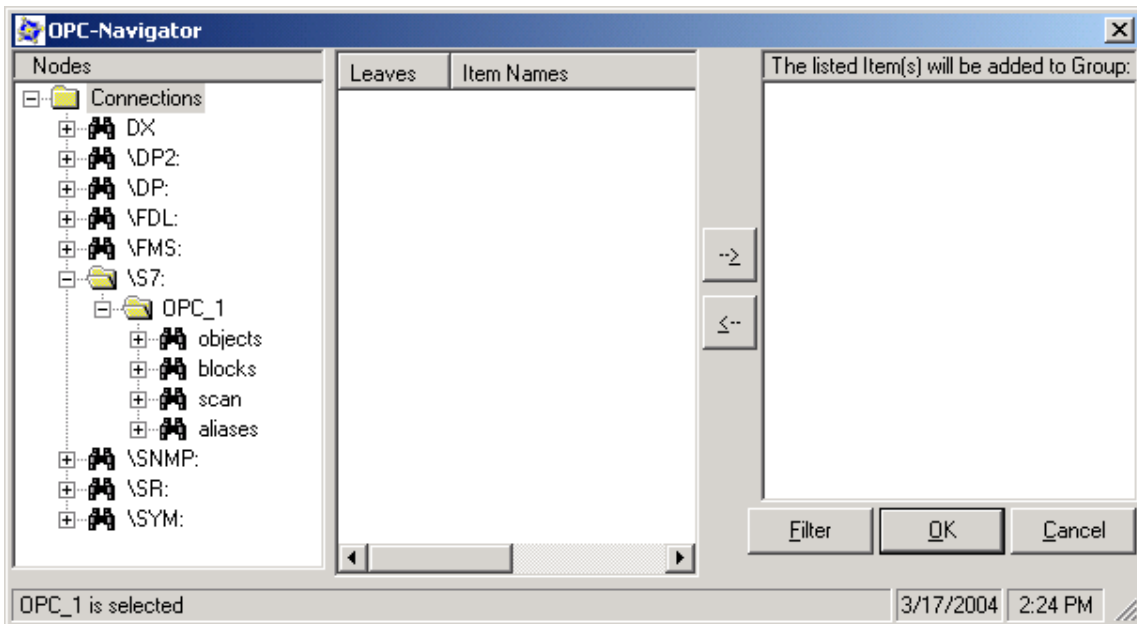
Note: This procedure describes how to use absolute addressing when configuring the OPC server. You can also use the STEP 7 symbol table for connecting the OPC server.

Use the following procedure to configure the OPC server to use an absolute address for accessing data in the controller.

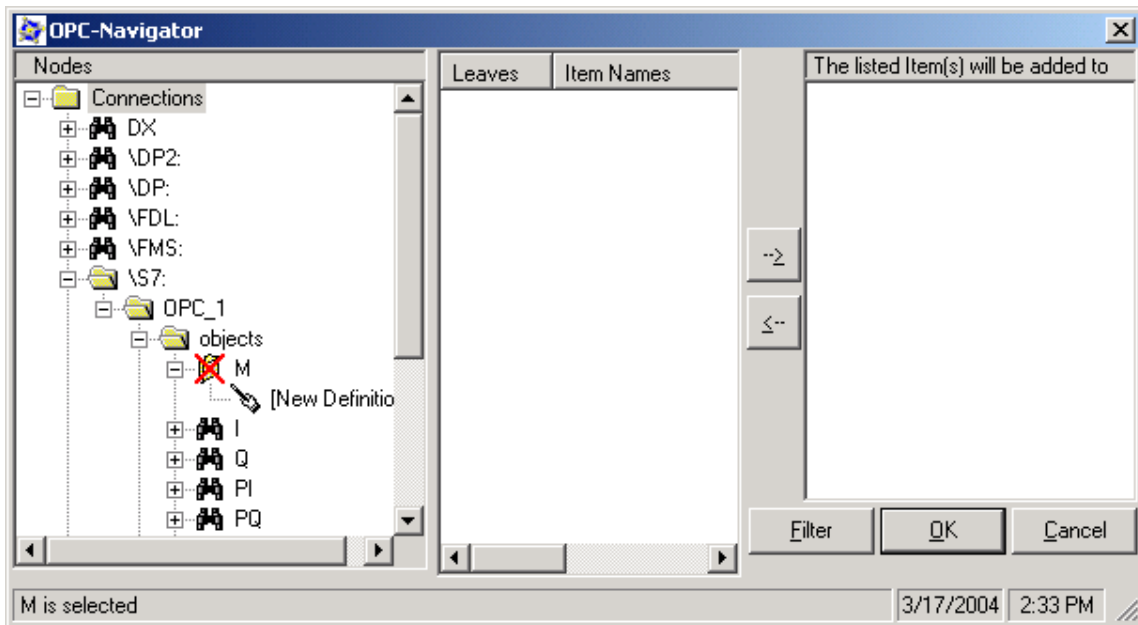
1. Open the OPC Navigator by double-clicking the connection (Group1) for the OPC server.



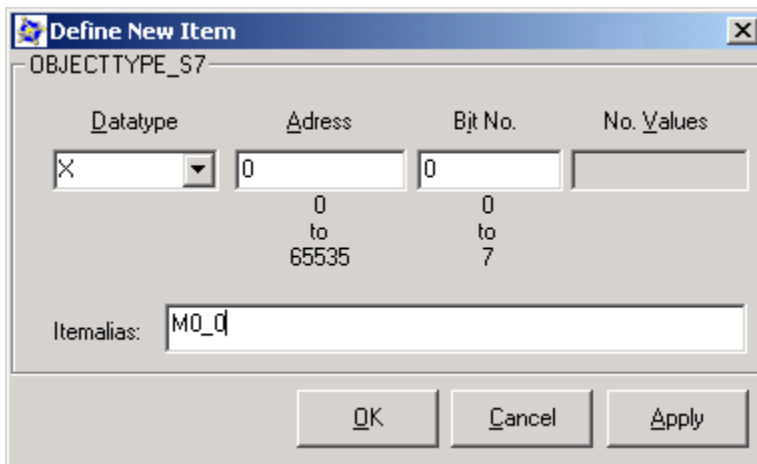
2. To add an item to be accessed, expand the \S7: folder and select OPC_1.



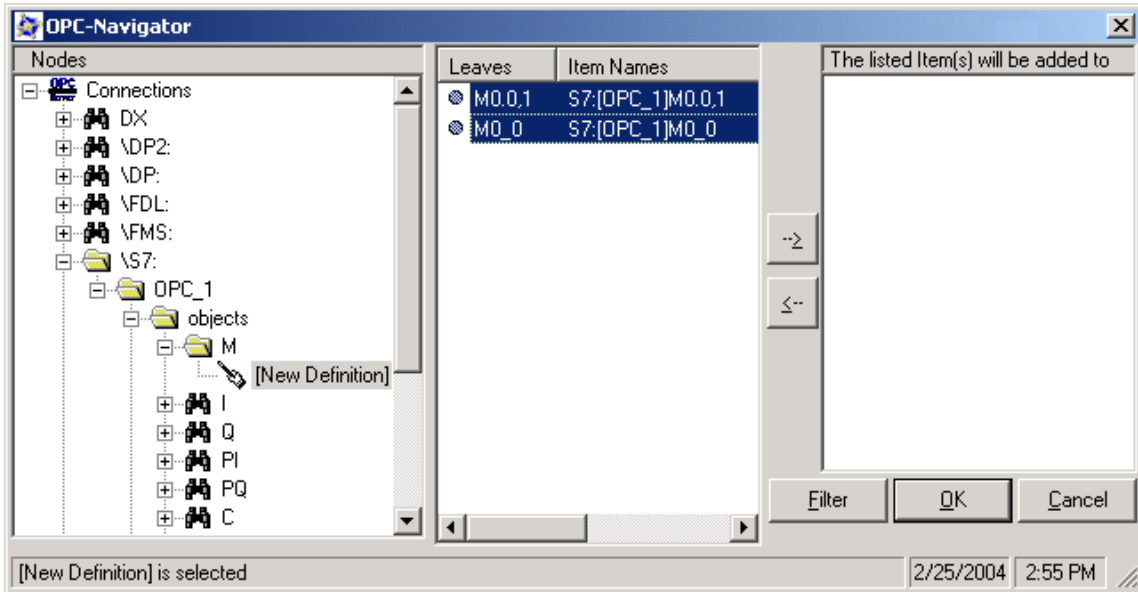
- To configure access to M 0.0, expand the Objects folder and expand the M folder (for the bit memory area).



- Double-click the New Definition icon to open the Define New Item dialog.
- To define a connection for M0.0, select X (for bit) field from the drop-down list in the Data Type and enter the byte address (0) and bit number (0). (You can also enter an alias for the item.)



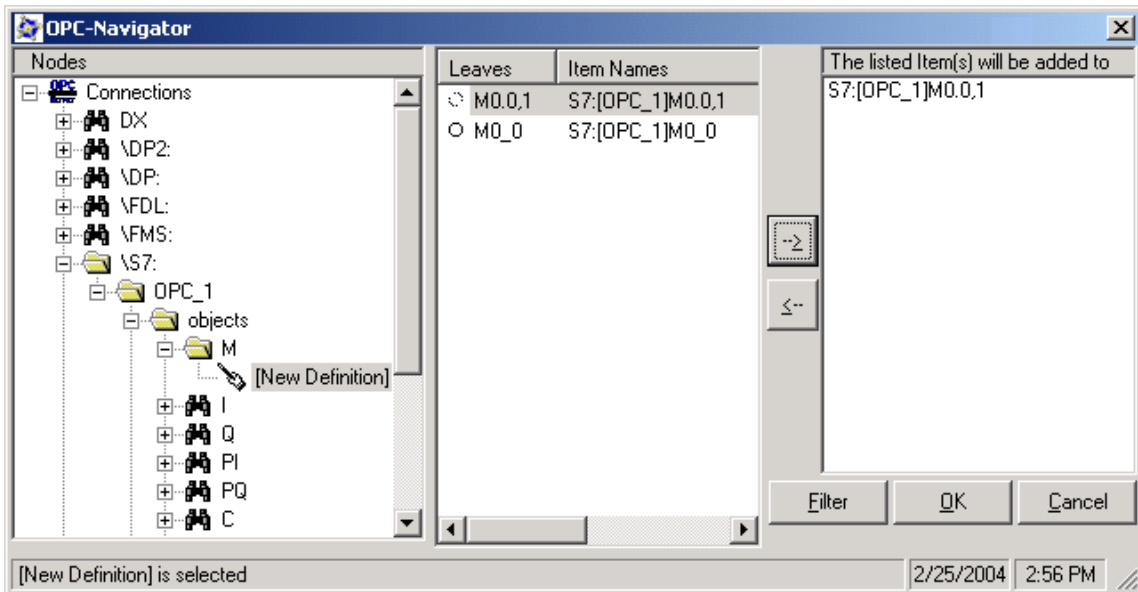
- Click OK to define an item for M0.0.



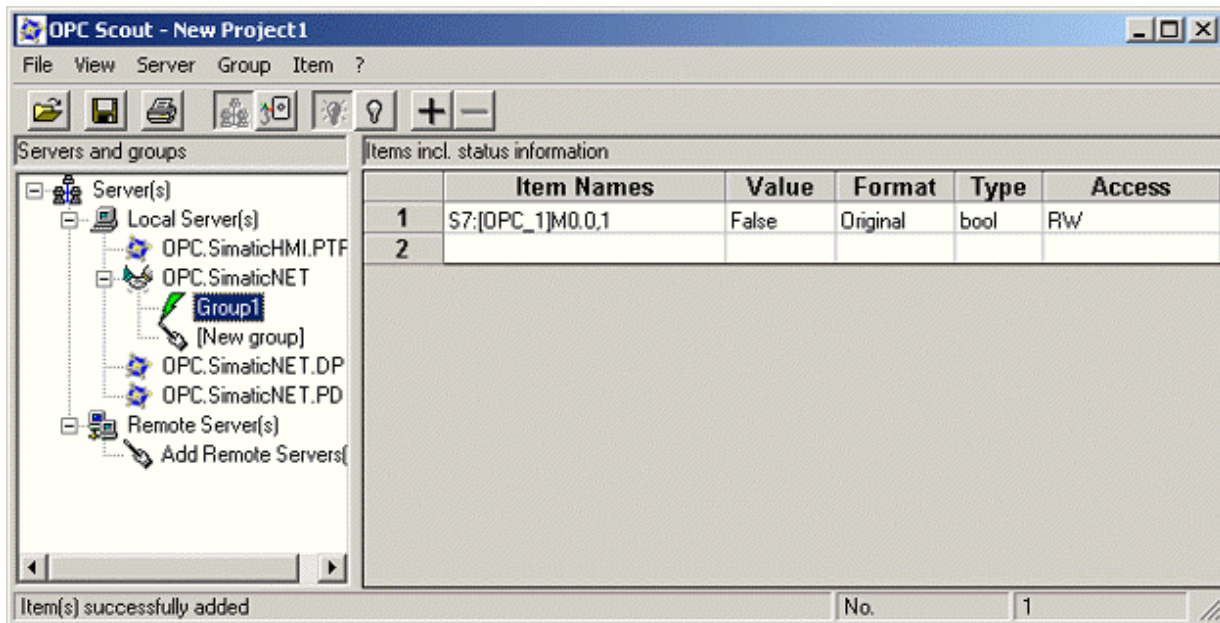
- Select the M0.0,1 entry and click the Add arrow (-->) to enter the following syntax that defines a connection for M0.0:

S7:[OPC_1]M0.0,1

- Select the entry (S7:[OPC_1]M0.0,1) and click OK to add the connection for M0.0 to Group1.



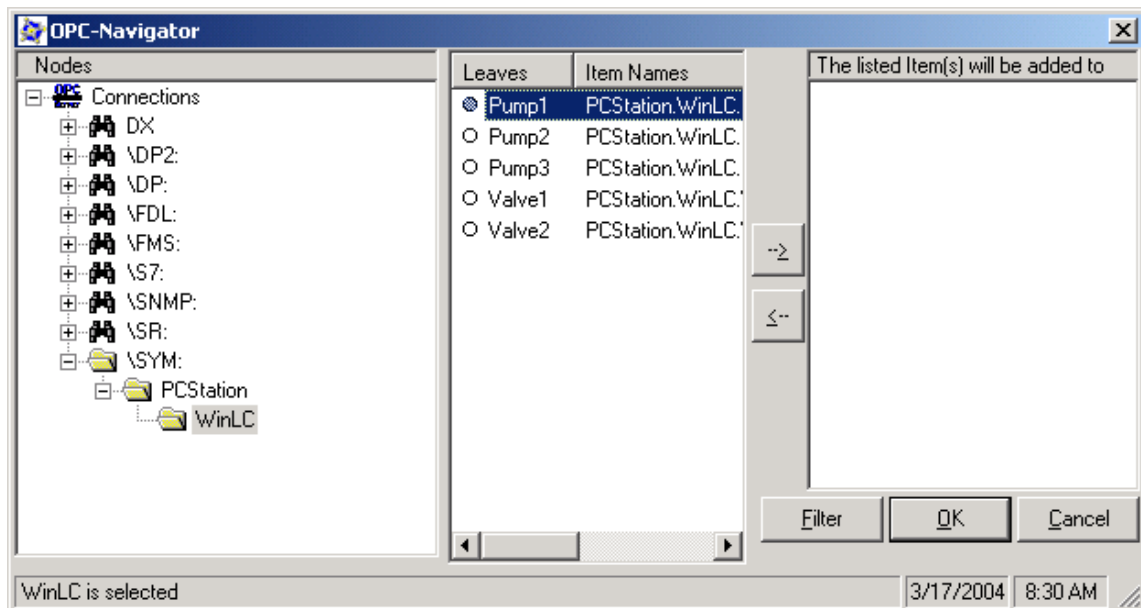
After adding the item to Group1, OPC Scout displays name and other parameters for the item. You can now use any of the methods supported by SIMATIC NET OPC server.



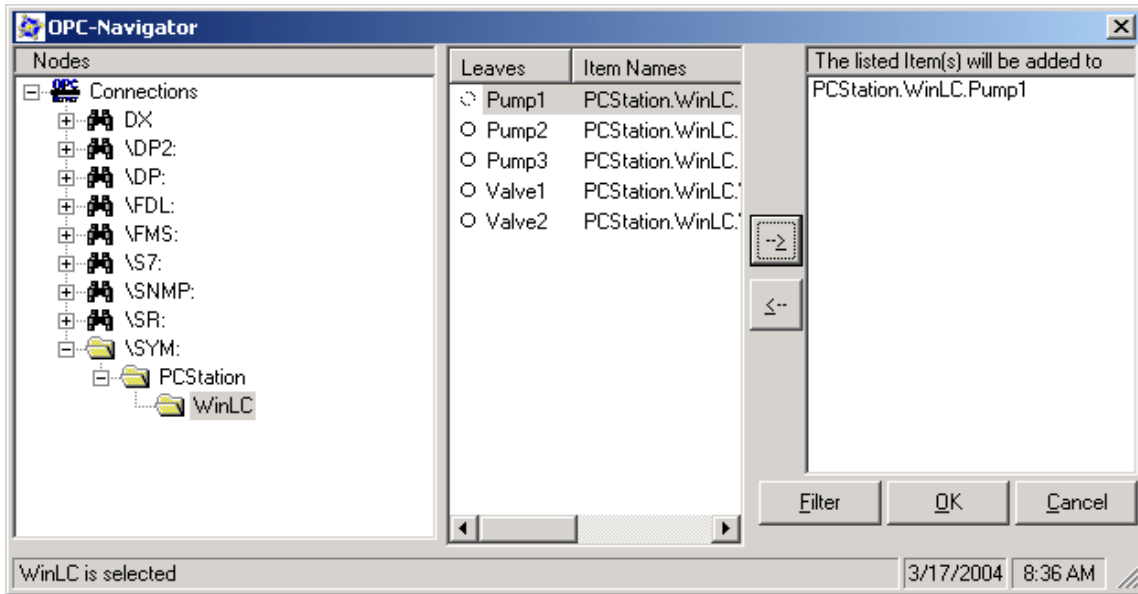
Configuring the Items to be Accessed (Using the STEP 7 Symbol Table)

If you created a symbol table for the STEP 7 program that you downloaded, you can use the symbols for connecting the OPC server to the data in the controller.

1. Open the OPC Navigator by double-clicking the connection (Group1) for the OPC server.
2. Browse to the folder for the controller to display the symbols that have been downloaded to the controller.

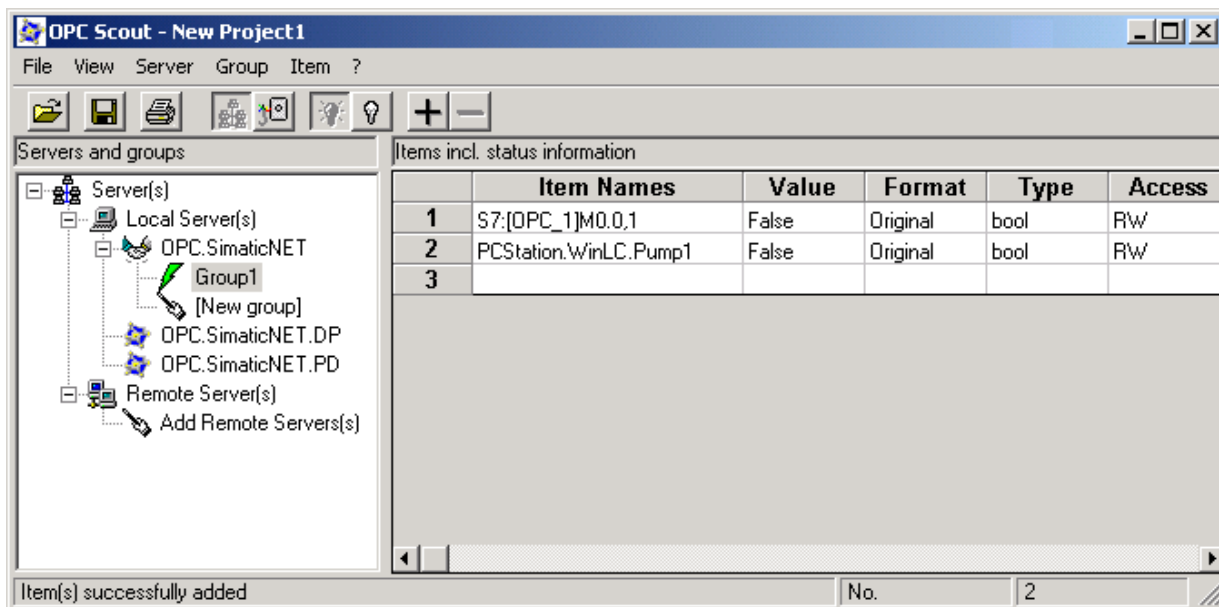


- After selecting the symbols for the data to be connected to the OPC server, click the Add button (-->).



- Click the OK button to add the symbol to Group1.

After adding the item to the group, OPC Scout displays symbol name and other parameters for the STEP 7 symbol.



Reference Information

Technical Data

Order Number

WinLC is a component of the WinAC package: 6ES7 671-0CC03-0YA0

Features

WinLC provides the following features:

- Accumulators: 4 (ACCU 1 to ACCU 4)
- Communications: PROFIBUS-DP master device
- Work memory: equal to the amount of physical memory in the computer
- Load memory: equal to the amount of memory (physical plus virtual) allowed by the Windows operating system
- Distributed I/O only, no local I/O: You can configure the size of the process-image I/O areas (I and Q memory areas) to be up to 8192 bytes. These memory areas can be accessed directly by the instructions in the control program. Using Load (L) and Transfer (T) instructions (for statement list) or the Assign Value (MOVE) instruction (for ladder logic) to the peripheral I/O (PI and PQ memory areas), you can access up to 16384 bytes of inputs and 16384 bytes of outputs.

WinLC communicates with the distributed I/O as a PROFIBUS-DP master device. As a master device, WinLC can communicate with up to 125 slave devices (either S7-DP slaves or other DP slaves).

Technical Specifications

The following table lists the technical information about WinLC.

WinLC	Description
Work memory	Limited to the amount of physical memory in the computer
Load memory	Limited to the amount of memory (physical plus virtual) allowed by the Windows operating system
Accumulators	4 (ACCU 1 to ACCU 4)
Local data	16 Kbytes per priority class (determined by HW Config, Memory tab)
Clock	Real-time system clock, based on the hardware clock of the computer
I/O (digital and analog)	16384 bytes total I/O, addressable over a range of 0 to 16383 You can freely assign the I/O between digital and analog inputs and outputs. For example, you can assign all of 16384 bytes to the inputs or all of the 16384 bytes to the outputs. However, the total amount allocated to all of the inputs and outputs cannot exceed the maximum of 16384 bytes.

WinLC	Description
Process image I/O (user configurable)	Inputs: 512 bytes (default) or configurable from 0 bytes to 8192 bytes (I 0.0 to I 8191.7) Outputs: 512 bytes (default) or configurable from 0 bytes to 8192 bytes (Q 0.0 to Q 8191.7)
Memory bytes	2 Kbytes
Retentive range (configurable)	Up to 2048 bytes (MB0 to MB2047)
Preset as retentive	16 bytes (MB0 to MB15)
Counters	512
Retentive range (configurable)	C0 to C511
Preset as retentive	8 (C0 to C7)
Timers	512
Retentive range (configurable)	T0 to T511
Preset as retentive	None
Clock memory	8 bits of clock memory (1 byte) 8 frequencies within 1 byte of bit memory (M): address is configurable
Address ranges for logic blocks (FB, FC, and DB):	FB0 to FB65535 FC0 to FC65535 DB1 to DB65535 (DB0 is reserved)
Number of connections	64
Nesting depth	24 per OB in a priority class (sequence layer) At any one time, a priority class can have one OB and up to two synchronous OBs (OB121 and OB122). Each OB in the priority class can have a nesting depth of 24.
Total number of blocks that can be downloaded to WinLC	No fixed limit: The number of blocks that can be downloaded is based on the memory requirements and the number of blocks in the program

**Caution**

Downloading a control program that is too large for the memory of the computer can lock up the computer or cause the operation of WinLC to become unstable. Non-responsive or non-deterministic operations can cause damage to equipment and/or injury to personnel.

Although STEP 7 and WinLC do not limit the number of blocks or the size of the control program, your computer does have a limit, based on the available disk space and RAM memory. The limit for the size of the control program and number of blocks for your computer can only be determined by testing a configured system against the requirements of your control application.

The following table lists specific information about the PROFIBUS-DP interface, as supported by WinLC.

PROFIBUS-DP interface	Description
DP address area	16384 bytes (inputs) and 16384 bytes (outputs)
Number of DP slaves supported for each submodule CP card	Dependent on the CP card <ul style="list-style-type: none"> • CP 5611: 32 • CP 5613: 125
Baud rate	Up to 12 Mbaud: 9.6 KBPS, 19.2 KBPS, 45.45 (31.25) KBaud, 93.75 KBPS, 187.5 KBPS, 500 KBPS, 1.5 MBPS, 3 MBPS, 6 MBPS, 12 MBPS
Baud rate search (as a DP slave)	Not applicable
Transfer memory (as a DP slave)	Not applicable
Maximum distance	Dependent on the baud rate

Execution Times

Execution Times of Instructions

The execution times listed in the two tables below (execution times for math operations and execution times for instructions) reflect the average execution times for STEP 7 programs running on WinLC. Actual execution times may vary, depending on your system.

Note: The execution times were measured on a SINUMERIK PCU50 computer (with a single 1.2 GHz Celeron processor). Tuning settings: 9000 microsecond execution time limit, 90% maximum execution load, and 1000 μ s forced execution sleep. Actual execution times may vary, depending on your computer.

Math Operation	Integer	Real	Double Word
Addition (+)	0.09 μ s	0.09 μ s	0.09 μ s
Subtraction (-)	0.09 μ s	0.09 μ s	0.10 μ s
Multiplication (*)	0.08 μ s	0.09 μ s	0.13 μ s
Division (/)	0.13 μ s	0.11 μ s	0.13 μ s

Instructions		Direct Addressing	Indirect Addressing	
Boolean Operations: A, AN, O, ON, X, XN	Memory areas:	I	0.04 μ s	0.05 μ s
		M	0.03 μ s	0.04 μ s
		L	0.03 μ s	0.05 μ s
		DB	0.03 μ s	0.05 μ s
		T	0.13 μ s	0.17 μ s
		C	0.05 μ s	0.05 μ s
Boolean operations (on the accumulator): ==I, <>I, >I, <I, >=I, <=I		0.04 μ s		
Operations on the bits of the status word: A==0, A<>0, A>0, A<0, A>=0, A<=0		0.04 μ s		

Instructions			Direct Addressing	Indirect Addressing
Transitional contacts	Edge Positive	FP	0.07 μ s	
	Edge Negative	FN	0.07 μ s	
Set/Reset operations (bit operands)	Set	S	0.06 μ s	0.04 μ s
	Reset	R	0.06 μ s	0.04 μ s
RLO Operations	Negate RLO	NOT	0.03 μ s	
	Set RLO	SET	0.01 μ s	
	Clear RLO	CLR	0.02 μ s	
	Save RLO	SAVE	0.03 μ s	
Operations on Timers	Pulse Timer	SP	0.17 μ s	0.17 μ s
	Reset timer	R	0.03 μ s	0.05 μ s
	Extended pulse timer	SE	0.17 μ s	0.17 μ s
	On-delay timer	SD	0.17 μ s	0.17 μ s
	Retentive on-delay timer	SS	0.17 μ s	0.17 μ s
	Off-delay timer	SF	0.20 μ s	0.21 μ s
Miscellaneous	Open DB	OPN	0.09 μ s	
	Load	L	0.04 μ s	
	Transfer	T	0.05 μ s	

Execution Time of DP Instructions

The table below lists the execution times for the SFCs used with the distributed I/O for WinLC.

Note: The execution times were measured on a SINUMERIK PCU50 computer (with a single 1.2 GHz Celeron processor). Tuning settings: 9000 microsecond execution time limit, 90% maximum execution load, and 1000 μ s forced execution sleep. Actual execution times may vary, depending on your computer.

SFC	Name	Description	CP 5613
SFC11	DPSYNC_FR	Synchronize groups of DP slaves	1.12 μ s
SFC13	DPNRM_DG	Reads the diagnostic data of a DP slave DP configuration tested: one ET 200M slave with one 8-input/8-output module and one 16-output module	2.20 μ s
SFC14	DPRD_DAT	Reads the consistent data from a DP slave	1.71 μ s
SFC15	DPWR_DAT	Writes the consistent data to a DP slave	1.87 μ s
SFC26	UPDAT_PI	Updates the process-image input table DP configuration tested: one ET 200M slave with one 8-input/8-output module and one 16-output module	0.83 μ s
SFC27	UPDAT_PO	Updates the process-image output table	16.30 μ s

Performance Test: Scan Time Jitter

To determine a sample of potential jitter in the scan cycle, the performance of WinLC was tested on a SINUMERIK PCU50 computer (with a single 1.2 GHz Celeron processor). Tuning settings: 9000 microsecond execution time limit, 90% maximum execution load, and 1000 μ s forced execution sleep.

The results of this test are provided only as an example of possible jitter. The performance of your application may vary, based on the configuration of your system and other parameters.

Based on the hardware configuration of your computer, and also the CPU utilization by other software applications, the scan cycle for WinLC can experience jitter. For example, different video cards and IDE hard drives can affect the specified performance of WinLC.

Configuration of WinLC for the Performance Test

To allow the maximum load on the CPU resources of the computer, WinLC was configured to run with the following parameters for the performance test.

Tuning Parameter	Value	Description
Priority	29	"Real-time/Critical"
Minimum sleep time	10 ms	Default settings
Minimum cycle time	0 ms	
Execution time limit	9000 μ s	Default settings
Forced execution sleep	1000 μ s	
Maximum execution load	90%	

Control Program Used for the Performance Test

For the performance test, WinLC executed a control program that performed the following operations during each scan cycle:

- The control program performed 3000 Boolean, 3000 Integer, and 3000 Floating Point operations.
- The control program read and wrote data to ET 200M I/O modules over a PROFIBUS-DP subnet that consisted of three nodes.
- The control program performed calculations to determine the distribution of performance data.

Other Applications That Were Executed During the Performance Test

While WinLC executed the test program, the computer also executed the following applications:

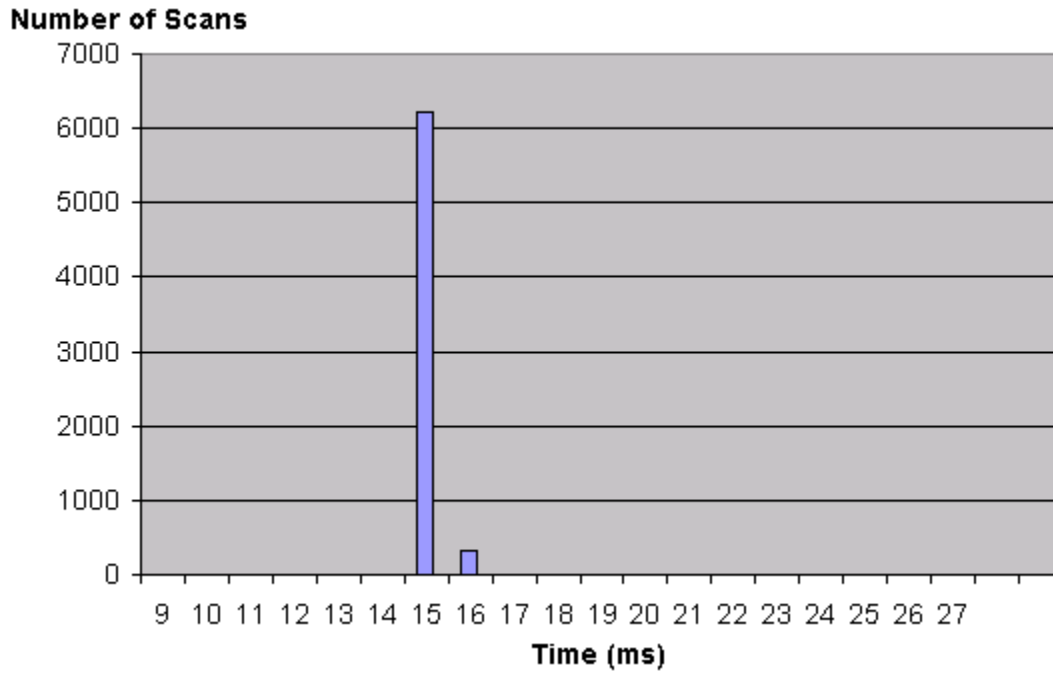
- STEP 7 monitored WinLC operations by updating 512 double-words of data in the Monitor and Modify Variables application.
- Two graphics-intensive test programs ran in infinite loops.
- Various Microsoft software applications (such as Word 2000, Excel 2000, Internet Explorer 5, and Outlook 2000) were opened, used, and closed.

Results of the Performance Test

As measured by the Windows Task Manager, the performance test produced the following results:

- The CPU utilization ranged between 99% and 100%.
- The average execution time of the WinLC program was 15 ms.

Actual performance may vary, depending on your system. The following figure provides a graphical summary of the potential jitter that could be experienced in the execution of a control program.



Troubleshooting

Troubleshooting Network Problems

WinLC helps you diagnose DP network problems. The control panel provides two status indicators (EXT1 and BUSF) that can be used to diagnose problems with the PROFIBUS-DP network. The table below describes the activity of the EXTF and BUSF indicators to help you determine the type of problem and a possible solution.

EXTF	BUSF	Description	Action
Off	Off	No configuration	Ensure that the DP configuration has been entered into your STEP 7 project. Download the project's System Data container to WinLC.
		Normal operation	The configured DP slaves are responding. No action is required.
On	Flashing	Station failure	Check to see that the bus cable is connected to WinLC (the CP card) and that all segments are correctly terminated at powered nodes. Check to see that the bus is not interrupted.
		At least one of the DP slaves could not be accessed	Wait for WinLC to complete the power-on cycle. If the indicator continues to flash, check the DP slaves or evaluate the diagnostic data for the DP slaves.
—	On	Bus fault (hardware failure)	Check the bus cable for an electrical short, or a broken wire or connection.
On	Off	Diagnostic error	Indicates that a fault condition has not been cleared or that a DP module with diagnostic capability has initiated OB82.

In addition to these visual indicators, you can use the Diagnose Hardware feature of the STEP 7 programming software to determine which nodes are experiencing problems and to determine the nature of the problem.

Responding to Diagnostic Events

If an error is detected by the controller, the error condition is logged in the diagnostic buffer as a diagnostic event. The diagnostic events that are typically associated with distributed I/O can cause the controller to execute the following OBs:

- OB40 responds to hardware interrupts (process alarms) generated by an I/O module with configured interrupt capability.
- OB82 responds to diagnostic interrupts generated by an I/O module with configured diagnostic interrupt capability.
- OB83 responds to module removal/insertion at a DP Slave, (for example, ET200M), which has been configured for module pull/plug support.
- OB85 responds to a priority class error. There are multiple causes for OB85 relating to the DP I/O system. If the controller attempts to copy a module's inputs to (or outputs from) the process image during the I/O cycle, and the module is not operational, then an OB85 is executed.
- OB86 responds to a station failure or some other interruption of the physical network (such as a short circuit).
- OB122 responds to an I/O access error by the user program. If OB122 is not programmed, the controller goes to STOP mode.

You can use SFC39 to SFC42 to disable, delay, or re-enable any of these OBs. If an OB is requested and the OB has not been downloaded to WinLC, the controller goes to STOP mode.

The local variables for these OBs contain startup information indicating the cause for executing the OB. The program for the OB can use this information for responding to the event. You can also use SFC13 (DPNRM_DG) to read the diagnostic information from a DP slave.

For information about using OBs and SFC13, see the online help for STEP 7 or the *System Software for S7-300/400 System and Standard Functions Reference Manual*. To view this manual from a computer where STEP 7 is installed, select the **Start > Simatic > Documentation > English** menu command and then double-click "STEP 7 - System and Standard Functions for S7-300 and S7-400".

System Status List (SSL)

Using SFC51 to Read the SSL

STEP 7 stores read-only information about the controller in the system status list (SSL) as a set of sublists.

You use SFC51 (RDSYSST) to access the entries in the SSL. You supply the input parameters SSL_ID and Index to access the records stored in the sublist. SFC51 returns a two-word header and a sublist or partial sublist. The header provides the following information about the sublist:

- The first word defines the length (size in bytes) of a record for the sublist.
- The second word defines the number of records contained in the sublist.

The requested information follows the header. The size of the sublist in bytes is the record length times the number of records.

Note: The SSL_ID and Index values are represented as hexadecimal (16#) numbers.

For more information about the system status list, see the online help for STEP 7 or the *System Software for S7-300/400 System and Standard Functions Reference Manual*. To view this manual from a computer where STEP 7 is installed, select the **Start > Simatic > Documentation > English** menu command and then double-click "STEP 7 - System and Standard Functions for S7-300 and S7-400".

WinLC supports the following SSL entries:

SSL ID 0000, 0100, 0200, 0300, 0F00	Interrupt and Error Assignment 0021, 0121, 0221, 0921, 0A21, 0F21	OB Start Events 0782
Module Identification 011, 0111, 0F11	Interrupt Status 0222	DP Master System 0090, 0190, 0F90
CPU Characteristics 0012, 0112, 0F12	Priority Class 0023, 0123, 0223, 0F23	Module Status 0A91, 0C91, 0D91, 0F91
Memory Areas 0013, 0113, 0F13	CPU Operating Mode 0124, 0424, 0524, 0F24	Rack and Station Status 0092, 0192, 0292, 0692
System Areas 0014, 0114, 0F14	Process Image Partitions 0025, 0125, 0225, 0F25	Expanded DP Master 0195, 0F95
Block Types 0015, 0115, 0F15	Communications Performance 0131	Diagnostic Buffer 00A0, 01A0, 0FA0
Local Module LED Status 0019, 0119, 0F19	Communications Status 0132, 0232	Module Diagnostics 00B1, 00B3, 00B4
Component Identification 001C, 011C, 0F1C	LED Status 0074, 0174, 0F74	

SSL_ID Descriptions

SSL_ID 0x00 (SSL ID)

0000, 0100, 0200, 0300, 0F00 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0000	Complete listing of the SSL IDs	All of the SSL IDs supported by the module
0100	SSL IDs for a specific SSL ID group	00yy, where yy = the SSL ID group: Lists the SSL IDs or the specified group of SSL IDs For example: 0100/0024 lists the indexes for the SSL ID 0x24 group (CPU Operating Mode) that were implemented by WinLC (0024, 0124, 0424, 0524, and 0F24)
0200	Valid (available) SSL ID	0zzz, where zzz = the SSL ID to be verified: <ul style="list-style-type: none"> If the SSL ID is valid, returns the index number If the SSL ID is invalid, returns a negative number
0300	All of the indexes for a specific SSL ID	0zzz, where zzz = the SSL ID: Lists all of the possible indexes for the specified SSL ID For example: 0300/0113 lists the indexes for SSL ID 0113 that were implemented by WinLC (0001, 0002, 0003, 0004, 0005, and 0006)
0F00	Header information only	Displays the number (quantity) of SSL IDs that were implemented by the module

SSL_ID 0x11 (Module Identification)

0011, 0111, 0F11 (hexadecimal)

Note: SSL_ID 0x11 does not provide information about submodules.

SSL_ID	Sublist	Index and Contents of the Record
0011	All of the information for a module	Order number, module type, version, and firmware version
0111	Specific information for a module	0001: Order number, module type, and version 0007: Firmware version
0F11	Header information only	

SSL_ID 0x12 (CPU Characteristics)

0012, 0112, 0F12 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0012	All characteristics for a module	MC7 processing unit, time system, system response, and MC7 language description
0112	One specific group of characteristics	0000: MC7 processing unit 0100: Time system 0200: System response 0300: MC7 language description
0F12	Header information only	

SSL_ID 0x13 (Memory Areas)

0013, 0113, 0F13 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0013	All of the memory areas for a module	Type, size, and other parameters for each memory area
0113	Specific memory area	0001: User memory 0002: Load memory integrated 0003: Load memory inserted 0004: Maximum insertable Load memory 0005: Backup memory 0006: Peer-to-peer memory (shadow memory)
0F13	Header information only	

SSL_ID 0x14 (System Areas)

0014, 0114, 0F14 (hexadecimal)

Note: Index 0008 displays the size of the bit memory (M) in bytes (instead of bits, as shown by index 0003), and index 0009 displays the size of the local memory (L) in Kbytes (instead of bytes, as shown by index 7). Use index 0008 and index 0009 to display information for controllers that have large memory capacity.

SSL_ID	Sublist	Index and Contents of the Record
0014	All system memory areas for a module	Size and other parameters for each area of system memory
0114	Specific area of system memory	0001: Input (I) memory area (in bytes) 0002: Output (Q) memory area (in bytes) 0003: Bit Memory (M) area (in bits) 0004: Timer (T) memory area (number of timers) 0005: Counter (C) memory area (number of counters) 0006: Peripheral input (PI) and peripheral output (PQ) memory areas (in bytes) 0007: Local (L) memory area (in bytes) 0008: Bit Memory (M) area (in bytes) 0009: Local (L) memory area (in Kbytes)
0F14	Header information only	

SSL_ID 0x15 (Block Types)

0015, 0115, 0F15 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0015	All block types for a module	Maximum number and size for each type of block
0115	Specific block type	0800: OB (number and size) 0A00: DB (number and size) 0B00: SDB (number and size) 0C00: FC (number and size) 0E00: FB (number and size)
0F15	Header information only	

SSL_ID 0x19 (Local Module LED Status)

0019, 0119, 0F19 (hexadecimal)

Note: SSL_ID 0x19 supports local, non-redundant CPUs. You can use SSL_ID 0x19 with a redundant H CPU only when the H CPU is in a non-redundant operating mode. Use SSL_ID 0x74 to access information for a redundant H CPU.

SSL_ID	Sublist	Index and Contents of the Record
0019	All of the LEDs for the local module	Status for all of the LEDs
0119	Specific LED for the local module	0002: INTF (Internal failure) 0003: EXTF (External failure) 0004: RUN (Run) 0005: STOP (Stop) 0006: FRCE (Force) 0008: BATF (Battery failure) 000B: BUSF1 (submodule 1 fault) 000C: BUSF2 (submodule 2 fault) 0012: BUSF3 (submodule 3 fault) 0013: BUSF4 (submodule 4 fault)
0F19	Header information only	

SSL_ID 0x1C (Component Identification)

0x1C (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
001C	All of the information for a component	Controller name, module name, module tag, copyright, serial number, project ID, module type, and manufacturer information
011C	Specific element for the component	0001: Name of the controller 0002: Name of the module 0003: Module tag 0004: Copyright entry 0005: Serial number 0007: Module type 0009: Manufacturer and profile identification
0F1C	Header information only	

SSL_ID 0x21 (Interrupt and Error Assignment)

0021, 0121, 0221, 0921, 0A21, 0F21 (hexadecimal)

SSL_ID)	Sublist	Index and Contents of the Record
0021	All of the OBs supported by the module	Priority class and OB number
0121	All possible OBs for a specific priority class	00: Free cycle (OB1) 0A (or 10 decimal): Time-of-Day OBs 14 (or 20 decimal): Time-Delay OBs 1E (or 30 decimal): Cyclic OBs 28 (or 40 decimal): Hardware OBs 32 (or 50 decimal): Communication and DP OBs 50 (or 80 decimal): Asynchronous Error OBs 64 (or 100 decimal): Startup OBs 78 (or 120 decimal): Synchronous error OBs
0221	Specific OB	OB number: Priority class and OB number
0921	All of the OBs for a specific priority class that have been downloaded to the module	00: Free cycle (OB1) 0A (or 10 decimal): Time-of-Day OBs 14 (or 20 decimal): Time-Delay OBs 1E (or 30 decimal): Cyclic OBs 28 (or 40 decimal): Hardware OBs 32 (or 50 decimal): Communication and DP OBs 50 (or 80 decimal): Asynchronous Error OBs 64 (or 100 decimal): Startup OBs 78 (or 120 decimal): Synchronous error OBs
0A21	All of the OBs that have been downloaded to the module	Priority class and OB number
0F21	Header information only	

SSL_ID 0x22 (Interrupt Status)

0022, 0122, 0222, 0822, 0922, 0F22 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0022	All of the OBs	All of the OBs that are supported by the module
0122	OBs for a specific priority class	Priority class: All of the OBs that are supported by the module for the specified priority class
0222	Start event for a specific OB	OB number: Start event and time for the requested OB
0822	OBs for a specific priority class being used by the module	Priority class: All of the OBs of the specific priority class that have been downloaded to the module
0922	All of the OBs being used by the module	All of the OBs that have been downloaded to the module
0F22	Header information only	

Note: For a list of the OBs supported by WinLC, refer to the following topics: Logic Blocks Supported by WinLC and Organization Blocks (OBs).

SSL_ID 0x23 (Priority Class)

0023, 0123, 0223, 0F23 (hexadecimal)

SSL_ID	Sublist	Contents of the Record
0023	All of the priority classes	Information about all of the priority classes supported by the module
0123	Specific priority class	Priority class: Information about that specific priority class
0223	Priority classes for the OBs being used by the module	Information about the priority classes of the OBs that have been downloaded to the module
0F23	Header information only	

SSL_ID 0x24 (CPU Operating Mode)

0124, 0424, 0524, 0F24 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0124	Operating mode transition	Last change of operating mode
0424	Current operating mode	Current operating mode of the module
0524	Specific operating mode	4520: Defective mode 5000: STOP mode 5010: STARTUP mode 5020: RUN mode 5030: HALT mode
0F24	Header information only	

SSL_ID 0x25 (Process Image Partitions)

0025, 0125, 0225, 0F25 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0025	All process image partitions	Process image partitions for all of the OBs that have been downloaded to the module
0125	Process image partition for a specific OB	Partition number: OB configured for that partition
0225	OBs assigned for a specific process image partition	OB number: Partition assigned for that OB
0F25	Header information only	

SSL_ID 0x31 (Communications Performance)

0131 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0131	Specific set of parameters	0001: Number of connections and baud rates 0002: Test and startup parameters 0003: Operator interface parameters 0004: Object management system (operating system function) 0005: Diagnostic functions and diagnostic entries 0006: Peer-to-peer performance parameters 0009: Number of run-time meters

SSL_ID 0x32 (Communications Status)

0132, 0232 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0132	Specific set of parameters	0001: Number and type of connections 0002: Connections configured 0003: Operator interface 0004: Protection level and mode switch selection 0005: Diagnostics 0006: Peer-to-peer status data 0008: Time system 000A: Baud rate
0232	Parameters for a redundant system (H CPU)	0004: Protection level and mode switch selection

SSL_ID 0x74 (LED Status)

0074, 0174, 0F74 (hexadecimal)

Note: Use SSL_ID 0x74 to access information about LEDs for any module, including a redundant H CPU module. See also SSL_ID 0x19.

SSL_ID	Sublist	Index and Contents of the Record
0074	All of the LEDs	Status for every LED of the module
0174	Specific LED	0002: INTF (Internal failure) 0003: EXTF (External failure) 0004: RUN (Run) 0005: STOP (Stop) 0006: FRCE (Force) 0008: BATF (Battery failure) 000B: BUSF1 (submodule 1 fault) 000C: BUSF2 (submodule 2 fault) 0012: BUSF3 (submodule 3 fault) 0013: BUSF4 (submodule 4 fault)
0F74	Header information only	

SSL_ID 0x82 (OB Start Events)

0782 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0782	Start event for each OB of a specific priority class that have been schedule to run but have not yet started	Priority class: Event, priority class, and OB number

SSL_ID 0x90 DP Master System

0090, 0190, 0F90 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0090	All DP masters configured on the network and downloaded to the module	DP master identifier, address, and attributes for all DP masters
0190	Specific DP master	DP master identifier: Address and attributes
0F90	Header information only	

SSL_ID 0x91 (Module Status)

0A91, 0C91, 0D91, 0F91 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0A91	All of the DP masters	DP master identifier, address, and module information for each DP master in the network configuration that was downloaded to the module
0C91	Specific module, identified by the logical base address	Logical base address: Features and parameters of the specified module
0D91	Specific station, identified either by rack/station, by DP master identifier, or by DP master identifier with station number	Station identifier: Features and parameters for all the modules of the specified station
0F91	Header information only	

SSL_ID 0x92 (Rack and Station Status)

0092, 0192, 0292, 0692 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0092	Expected status of the stations of a DP master	0: Local DP master DP master identifier: Specific DP master
0192	Configuration and activation status for the stations of a DP master	0: Local DP master DP master identifier: Specific DP master
0292	Actual status for the stations of a DP master	0: Local DP master DP master identifier: Specific DP master
0692	OK state for the stations of a DP master	0: Local DP master DP master identifier: Specific DP master
0F92	Header information only	

SSL_ID 0x95 (Expanded DP Master System)

0195, 0F95 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
0195	Specific DP master	DP master identifier: Properties for the stations of the specified DP master (such as DP mode, equidistant mode and cycle, clock synchronization, and transmission rate)
0F95	Header information only	

SSL_ID 0xA0 (Diagnostic Buffer)

00A0, 01A0, 0FA0 (hexadecimal)

SSL_ID	Sublist	Index and Contents of the Record
00A0	All of the entries in the diagnostics buffer	Event information for every event listed in the diagnostics buffer
01A0	Most recent entries in the diagnostics buffer	Number: Event information for the specified number of entries in the diagnostics buffer
0FA0	Header information only	

SSL_ID 00B1, 00B3, and 00B4 (Module Diagnostics)

00B1, 00B2, 00B4 (hexadecimal)

Note: The information varies according to the type of module specified.

SSL_ID	Sublist	Index and Contents of the Record
00B1	Diagnostic information (4 bytes) for a specific module, identified by the logical base address	Logical base address: First 4 bytes of the diagnostic information
00B3	All of the diagnostic information for a specific module, identified by the logical base address	Logical base address: Complete diagnostic information
00B4	Specific DP slave, identified by the configured diagnostic address	Diagnostic address: Standard diagnostic information for a DP station

Glossary

C

Cold Restart: The controller executes OB102 before starting the free cycle (OB1). Like a warm restart, a cold restart resets the peripheral inputs (PI) and changes the peripheral outputs (PQ) to a pre-defined safe state (default is 0). However, a cold restart does not save the retentive memory (M, T, C, or DB), but sets these areas to their default (initial) values.

Control Program: The control program is the application program created with STEP 7 and downloaded to the controller for execution. The control program includes all organization blocks (such as OB1 or OB35) and the other logic blocks that they call, including functions (FCs), system functions (SFCs), function blocks (FBs), and system function blocks (SFBs).

E

Execution Monitor: The execution monitor of the controller measures the time that the controller sleeps and ensures that the controller does not exceed the maximum execution load. The execution monitor uses the maximum execution load and the execution time limit to calculate the forced execution sleep time.

Execution Time: The execution time is the actual time the controller takes to complete one pass through the instructions of the control program. This includes executing OB1 and updating the I/O.

Execution Time Limit: The execution time limit defines the maximum amount of time allowed for the controller to execute the control program. The execution monitor uses this value and the maximum execution load to calculate the forced execution sleep time.

F

Forced Execution Sleep Time: This read-only field shows how much sleep time (in microseconds) is required during the monitor interval to meet the maximum execution load requirement.

Free Cycle: The free cycle consists of the basic tasks for priority class 1: writing to the outputs, reading the inputs, executing OB1, and completing the sleep time requirement before triggering the next free cycle. The controller executes these tasks at the base, or lowest, internal priority level for executing the OBs. (Priority level in this context refers to OB priority classes, not the operating system priority level.)

J

Jitter: Jitter is the difference in the actual scan cycle time from the configured minimum scan time.

M

Maximum Execution Load: The maximum execution load is the maximum percentage of CPU usage that is allocated for the controller. The execution monitor uses this value and the execution time limit to calculate the forced execution sleep time.

Minimum Cycle Time: The minimum cycle time is the minimum number of milliseconds from the start of one cycle to the start of the next cycle. You enter a value for the minimum cycle time when you use STEP 7 to configure the system data for the controller. You can use the tuning panel to adjust this value as you test the performance of the controller. After you have tuned the performance of the controller, use STEP 7 to enter the optimum cycle time value and download the new system data. Any value for the cycle time that you enter with the tuning panel is overwritten by the value in the system data when the controller changes from STOP mode to RUN mode.

Minimum Sleep Time: The minimum sleep time is the specific amount of time that the controller must wait before starting the next scan cycle. You use the tuning panel to configure this parameter. The controller uses the minimum sleep time and the minimum cycle time parameters to calculate the start of the next scan cycle.

Monitor Interval: The length of time used by the execution monitor in determining whether to add a forced sleep time. The monitor interval is the sum of the execution time limit and the forced execution sleep time that is calculated based on the maximum execution load percentage.

O

Organization Block (OB): The OBs represent the interface between the operating system and the control program. Called by the operating system, they control cyclic and interrupt-driven program execution, startup behavior of the controller and error handling.

P

Priority: The priority of an application determines the order in which the operating system executes or interrupts an application in relation to the other applications that are running on the computer. An application with a higher priority interrupts and suspends the execution of an application with a lower priority. After the application with the higher priority finishes, the application with the lower priority resumes. A higher number indicates a higher priority.

Priority Class: The priority class determines the order in which the controller executes the individual sections of the control program. Organization blocks (OBs) are ranked by priority class. Higher priority OBs interrupt lower priority OBs. The free cycle (OB1) has the lowest priority. You can use STEP 7 to change the priority class for an OB. A higher number indicates a higher priority class.

R

Restart Method: The restart method determines which startup OB is executed whenever the controller changes from STOP mode to RUN mode. The startup OB allows you to initialize your control program and variables. The two restart methods are Cold Restart (OB102) and Warm Restart (OB100).

S

Scan Cycle: The scan cycle includes writing to the outputs, reading the inputs, executing OB1 and all other OBs, and completing the sleep time requirement.

Scan Cycle Time: The scan cycle time is the time required to execute the complete scan cycle, which includes the execution of OB1 and the minimum sleep time.

Sleep Time: The sleep time is the difference between the execution time of the free cycle and the total scan time. Sleep time measures the time between the completion of OB1 and the start of the next scan cycle, and ensures that the next scan cycle does not start until the end of the sleep interval. However, if the start event for an interrupt OB (such as OB40) occurs during the sleep time, the controller executes that OB.

System Function (SFC): An SFC is a preprogrammed function that is integrated as a part of the operating system of the controller and is not downloaded as part of the control program. You can call the SFC in your control program. Like a function (FC), an SFC is a block "without memory."

System Function Block (SFB): An SFB is a function block that is integrated as a part of the operating system of the controller and is not downloaded as part of the control program. Like a function block (FB), an SFB is a block "with memory." You must also create an instance data block (DB) for the SFB. The instance DB is then downloaded to controller as part of the control program.

W

Wait Time: The wait time, or sleep time, is the time that the controller is not using the CPU. During this time, the operating system can run other applications.

Warm Restart: The controller executes OB100 before starting the free cycle (OB1). A warm restart resets the peripheral inputs (PI) and changes the peripheral outputs (PQ) to a pre-defined safe state (default is 0). The warm restart also saves the current value for the retentive memory areas for the memory bits (M), timers (T), counters (C), and data blocks (DBs).

Index

A

- Above-normal priority, 91
- Access points, 24
- Access verification dialog, 46
- Accessing WinLC from STEP 7, 51
- Adding blocks to Load memory with SFC82 and SFC84, 44
- Adding sleep time, 100, 101, 107
- Addresses, 62, 68
 - diagnostic, 68
 - distributed I/O, 62
 - logical addresses, 62
- Adjusting, 97
 - minimum scan cycle time, 97
 - minimum sleep time, 97
 - priority, 83, 90, 91
 - sleep-monitoring algorithm, 97, 101
- Advanced tuning, 35
- Alarms, 138
- All indicators flashing, 33
- Always on top option, 45
- Archiving, 39
- Assigning addresses, 62, 65, 68
- Assigning names, 9
- Asynchronous threads, 90
- Authorization, 10, 11
 - installing, 11
 - removing, 11
- AuthorsW program, 11
- Autostart, 42
- Autostart CPU, 45
- Avoiding jitter, 100, 101, 107

B

- Background priority, 91
- Basic tasks, 13

- BATF status indicator, 33
- Battery fault, 33
- Baud rate, 129
- Blue Screen, 2, 40, 42, 69, 72
- BRCV, 70, 80
- BSEND, 70, 80
- BUSF1 status indicator, 33, 137
- BUSF2 status indicator, 33

C

- Change Password dialog, 47
- Changing, 30, 90
 - CPU keyswitch position, 30
 - mode buttons, 30
 - operating mode, 30
 - password, 47
 - priority, 90
 - sleep-monitoring algorithm, 101, 107
- Characters (invalid) for instance name, 28
- Clock, 82
- Clock memory, 52, 55
- Closing the control panel, 29
- Cold restart, 42, 49, 54, 72
 - retentive memory, 58
- Commands
 - Diagnostic buffer, 37
 - MRES (memory reset), 32
 - Options
 - Customize, 45
 - Security, 46
 - Tuning panel, 35
- Commissioning wizard, 9
- Communication, 1, 13, 17, 19, 24, 70, 71
 - commissioning the CP card, 9
 - DPV1 extensions, 71
 - getting started tasks, 13

- peer-to-peer, 70
 - with I/O devices, 17, 19
- Computer requirements, 5
- Configuration console, 9
- Configuring, 13, 20, 26, 52, 68
 - clock memory, 55
 - communications, 24
 - controller communications, 13
 - CP card as submodule, 17, 19
 - CP card in PC station, 9
 - cycle/clock memory, 52
 - cyclic interrupt, 52, 61
 - diagnostics/clock, 52
 - DP addresses, 68
 - interrupts, 60
 - memory, 52
 - operational parameters, 52
 - PC station, 14
 - project in STEP 7, 26
 - retentive memory, 52, 58
 - scan cycle, 56
 - startup, 52
 - startup characteristics, 54
 - submodule, 20
 - time-of-day interrupt, 52, 59
- Connecting to the OPC server, 111
- Context-sensitive help, 6
- CONTROL, 70, 80
- Control panel, 1
 - opening and closing, 29
 - status indicators, 33
- Control program, 26, 32
 - archiving, 39
 - changing keyswitch position, 30
 - deleting, 32
 - downloading, 26
 - restoring, 39
- Controller, 13, 20, 26, 40, 42, 86
 - autostart, 45
 - configuring communications, 13, 24
 - configuring in PC station, 14
 - configuring in STEP 7, 26
 - invalid characters, 28
 - memory reset changes, 32
 - naming, 28
 - restart method, 49
 - scan cycle, 83
 - setting priority, 86, 90, 91
 - shutting down, 29, 40, 42
 - starting, 29, 42, 48
 - startup, 40
 - status indicators, 33
 - submodule, 17, 20
 - system status list (SSL), 139
- Correction factor for clock, 52
- CP 5613 card, 26
- CP Card, 9, 13, 14, 16, 17, 19, 20, 24
 - commissioning, 9
 - configuring as submodule, 14, 17, 19, 20
 - configuring in STEP 7, 26
 - PC station component, 16
- CPU indicators, 33
- CPU menu
 - diagnostic buffer, 37
 - MRES (memory reset), 32
 - options
 - customize, 45
 - security, 46
 - tuning panel, 35
- CPU usage, 35, 86
 - jitter, 86
- Creating, 39
 - archive file, 39
 - password, 46

- submodule, 20
- Customize command (CPU menu), 45
- Cycle time, 35, 56, 83, 97, 101, 107
- Cycle/clock memory, 35, 52, 55, 56
- Cyclic interrupt, 52, 61, 72
- D**
- Dashes (in the controller name), 28
- Data retention, 42
- Defective state, 33
- Deterministic scan cycle, 91
- Diagnosing hardware, 137
- Diagnostic addresses for DP I/O, 68
- Diagnostic buffer, 37, 40, 138
 - saving the contents, 40
- Diagnostic events, 37, 138
- Diagnostic interrupts, 72, 138
- Diagnostics/clock, 52
- Display language, 45
- Distributed I/O, 1, 129
- Downloading a control program, 26
- DP instructions, 134
- DP network, 62, 137
 - assigning addresses, 62
 - troubleshooting, 137
- DPV1 extensions, 71
- E**
- Edit component dialog, 20
- Eliminating forced sleep interval, 107
- English language option, 45
- Error OBs, 72
- Errors, 33, 138
- Events, diagnostic, 37
- Execution, 59, 91
 - adjusting priorities, 90, 91
 - OB10, 59
- Execution monitor, 35, 83, 86, 93, 97, 101, 107
- Execution time, 35, 56, 75, 80, 83, 97, 107, 132, 134
- DP instructions, 134
- instructions and math operations, 132
- SFBs, 80
- SFCs, 75
- Execution time limit, 101
- External power supply, 44
- EXTF status indicator, 33
- F**
- Features of WinLC, 2
- File menu
 - Archive command, 39
 - Restore command, 39
- Forced execution sleep time, 93, 101, 107
- Format for diagnostic buffer, 37
- Forward slash (illegal character), 28
- FRCE status indicator, 33
- Free cycle, 83, 86, 97, 101, 107
- French language option, 45
- Full-text search, 6
- G**
- German language option, 45
- GET, 70, 80
- Getting started, 6, 13
- H**
- Hardware configuration, 14, 26, 51, 68
 - clock memory, 55
 - cyclic interrupt, 61
 - interrupts, 60
 - OBs, 72
 - operational parameters, 52
 - retentive memory, 58
 - scan cycle, 56
 - startup characteristics, 54
 - time-of-day interrupt, 59
- Hardware interrupts, 60, 72, 138
- Help menu

Using help, 6
Help on diagnostic buffer, 37
Hyphen in the controller name, 28

I

I/O access error, 72, 138
I/O device communication, 17
IF slot, 17, 19, 20
Index (PC station slot), 16, 20
Indicator lights, 33
Inserting sleep time, 100, 101, 107
Installation requirements, 5
Installing, 10, 11
 Privilege requirements, 6
 SIMATIC NET, 9
 WinAC authorization, 11
 WinAC components, 9, 10
Instance, invalid characters, 28
Instructions, 132, 134
Interface slots, 19
Interrupt OBs, 72, 86
Interrupts, 60, 61, 83, 138
INTF status indicator, 33
Invalid characters for controller names, 28

J

Jitter, 35, 86, 90, 91, 93, 100, 101, 107
 performance testing, 135
 reducing, 35, 91, 93, 100, 101, 107
 tuning panel, 35

K

Keyswitch position, 30

L

Language selection, 45
LED indicators, 1, 33
Load memory, 40, 42, 44
 Adding blocks with SFC82 and SFC84, 44
Logic blocks, 69
 OBs, 72

SFBs, 80
SFCs, 75
Logical address, 65

M

Main program cycle, 72
Managing sleep time, 93
Math operations, 132
Maximum execution load, 35, 93, 101, 107
Memory, 40, 52, 129
 configuring, 52
 requirements, 5
 retentive, 58
 saving and restoring, 40
 specifications, 129
Memory areas, 32
Meter, run-time, 82
Minimum scan cycle time, 35, 56, 83, 93, 97, 135
Minimum sleep time, 35, 83, 93, 97, 100, 101, 107
Minus sign (in the controller name), 28
Mode Buttons, 30
Module removal/insertion, 138
MPI, 24
MRES, 1, 32, 33

N

Names, 9, 14
 assigning, 9
Naming an instance of the controller (invalid characters), 28
Network, 137
 protocols, 24
 STEP 7 communications, 24, 26
 troubleshooting, 137
Normal priority, 91

O

OB1, 49, 83, 86, 97, 101, 107
OB10, 52, 59
OB100, 49, 54, 83, 101

- OB100 and OB102, 42, 54
- OB102, 49, 54, 101
- OB122, 138
- OB20, 52, 60, 97
- OB32 to OB36, 52, 61, 107
- OB35, 52, 61, 83, 86, 97, 101, 107
- OB40, 52, 83, 86, 97, 101, 138
- OB61, 101
- OB80, 61
- OB80 to OB87, 60
- OB82, 52, 138
- OB83, 52, 138
- OB85, 52, 101, 138
- OB86, 52, 138
- OBs, 72
 - execution, 35, 86
 - supported by WinLC, 69
- ON status indicator, 33
- OPC server, 111
- Opening the control panel, 29
- Operating mode, 30, 45
 - at startup, 45
 - RUN, 30
 - RUN-P, 30
 - status indicators, 33
 - STOP, 30
 - User privileges required, 6
- Operating system threads, 90
- Operational parameters, 52
- Optimizing performance, 35
- Order number, 129
- Organization blocks, 72
- Overview, 1
- P**
- Panel, 1
 - opening and closing, 29
 - status indicators, 33
- Part number, 129
- Password, 46, 47
 - changing, 47
 - creating, 46
- PC station, 9, 14, 20, 26
 - configuring with SIMATIC NET, 14
 - configuring with STEP 7, 26
 - CP card as component, 9
 - CP card as submodule, 17, 20
 - OPC server, 111
- PC-based controller, 1
 - scan cycle, 83
 - starting, 29
- PCI slot, 16
- Peer-to-peer communications, 24, 70
- Performance, 83, 91, 93, 97, 101
 - test for jitter, 135
 - tuning, 35, 83, 91, 93, 97, 101
- Personal computer, 5
- PG/PC interface, 24
- Power failure, 58
- Power supply, 44
- Power-Down State, 40
- Priority, 35, 72, 83, 91, 93, 97, 101, 107
 - adjusting, 35, 83, 90, 91, 93
 - cyclic interrupts, 61
 - interrupt OBs, 60
 - OBs, 72
 - setting, 86, 107
- Priority class error, 138
- Privileges, 6
- Product overview, 1
- PROFIBUS-DP, 1, 20, 24, 129
 - assigning addresses, 62
 - communicating with I/O, 17
 - CP card submodule, 20
 - DPV1 extensions, 71

- logical addresses, 65
- network troubleshooting, 137
- specifications, 129
- Programming error, 72
- Protocols, 24
- PUT, 70, 80
- R**
- Rack failure, 72
- RAM requirements, 5
- RDSYSST (SFC51), 139
- Real-time priority (Windows), 91
- Real-time system clock, 82
- Recovering from a defective state, 33
- Register Controller for Start at PC Boot Command (CPU menu), 48
- Removing, 11
 - Authorization, 11
- Removing WinAC software, 12
- Renaming the controller (invalid characters), 28
- Requirements, 5
- Resetting memory areas, 32
- Resources (computer), 97
- Responding to diagnostic events, 138
- Restart, 42, 54
 - autostart feature, 45
 - characteristics, 54
 - retentive memory, 58
 - selecting method, 49
- Restoring, 39
- Retentive memory, 52, 58
- Ring ON button, 20
- RUN, 30
- RUN status indicator, 33
- RUN-P, 30
- Run-time meter, 82
- S**
- Scan cycle, 35, 56, 83, 86, 91, 93, 97, 101, 107, 135
- configuring, 56
- jitter, 86, 135
- monitoring, 35
- SDB0, 52
- Security, 46
 - Security command (CPU menu), 46
 - setting level, 46
- Selecting, 45
 - autostart, 45
 - language, 45
 - restart method, 49
- Service, 48
 - starting the controller after reboot, 48
- Setting, 46
 - PG/PC Interface, 24
 - priority, 90, 91, 107
 - security options, 46
- Setup, 9, 10
- SFBs, 80
 - execution times, 80
 - peer-to-peer communications, 70
 - supported by WinLC, 69
- SFC0, 82
- SFC1, 82
- SFC11, 134
- SFC13, 134, 138
- SFC14, 65, 134
- SFC15, 65, 134
- SFC22, 58
- SFC26, 134
- SFC27, 134
- SFC39, 138
- SFC42, 138
- SFC47 (WAIT), 83, 93, 97, 100, 101, 107
- SFC51, 139
- SFC62, 70
- SFC64, 82

- SFC82, 42, 44
- SFC83, 42, 44
- SFC84, 42, 44
- SFCs, 75
 - execution times, 75
 - reading the system status list, 139
 - SSL IDs supported, 139
 - supported by WinLC, 69
- Shutting down, 29, 40, 42, 44
- SIMATIC NET, 2, 9, 14
 - OPC server, 111
- Slash (illegal character), 28
- Sleep management techniques, 93
- Sleep time, 35, 83, 86, 93, 97, 100, 101, 107
- Sleep-monitoring algorithm, 83, 93, 97, 101, 107
- Specifications, 129
- SSL ID, 139
 - 0x00, 140
 - 0x11, 140
 - 0x12, 141
 - 0x13, 141
 - 0x14, 142
 - 0x15, 142
 - 0x19, 143
 - 0x1C, 143
 - 0x21, 144
 - 0x22, 145
 - 0x23, 145
 - 0x24, 146
 - 0x25, 146
 - 0x31, 147
 - 0x32, 147
 - 0x74, 148
 - 0x82, 148
 - 0x90, 148
 - 0x91, 149
 - 0x92, 149
 - 0x95, 150
 - 0xA0, 150
 - 0xB1, 150
 - 0xB3, 150
 - 0xB4, 150
- assignment of interrupts and errors (0x21), 144
- available SSL IDs, 139
- available SSL IDs (0x00), 140
- backup memory (0x13), 141
- block types (0x15), 142
- C memory size (0x14), 142
- communications performance (0x31), 147
- communications status (0x32), 147
- component identification (0x1C), 143
- contents (SSL IDs supported), 139
- CPU characteristics (0x12), 141
- CPU LED status (0x74), 148
- CPU LED status, local only (0x19), 143
- CPU operating mode (0x24), 146
- DB number and size (0x15), 142
- diagnostic buffer (0xA0), 150
- DP master system (0x90), 148
- DP master system, expanded (0x95), 150
- DP module diagnostics (00B1, 00B3, 00B4), 150
- DP module status (0x91), 149
- DP rack/station status (0x92), 149
- error assignment (0x21), 144
- events that start OBs (0x82), 148
- expanded DP master system (0x95), 150
- FB and FC number and size (0x15), 142
- H CPU LED status (0x74), 148
- I memory size (0x14), 142
- identification, module (0x11), 140
- indexes for an SSL ID (0x00), 140
- interrupt and error assignment (0x21), 144
- interrupt status (0x22), 145
- L memory size (0x14), 142

- LED status for redundant modules (0x74), 148
- LED status, local only (0x19), 143
- list SSL IDs (0x00), 140
- Load memory (0x13), 141
- local CPU LED status (0x74), 148
- local module LED status (0x19), 143
- M memory size (0x14), 142
- master system (0x90), 148
- maximum number and size of blocks (0x15), 142
- memory area sizes (0x14), 142
- memory areas (0x13), 141
- mode, CPU (0x24), 146
- module diagnostics (00B1, 00B3, 00B4), 150
- module identification (0x11), 140
- module LED status, local and redundant CPU (0x74), 148
- module LED status, local only (0x19), 143
- module status (0x91), 149
- OB number and size (0x15), 142
- OB priority class (0x23), 145
- OB start events (0x82), 148
- operating mode (0x24), 146
- order number (0x11), 140
- peer-to-peer memory (0x13), 141
- performance, communications (0x31), 147
- PII memory size (0x14), 142
- PIQ memory size (0x14), 142
- priority class (0x23), 145
- process image partitions (0x25), 146
- Q memory size (0x14), 142
- rack/station status (0x92), 149
- redundant CPU LED status (0x74), 148
- SDB number and size (0x15), 142
- SFC51, 139
- shadow memory (0x13), 141
- size of blocks (0x15), 142
- size of the memory areas (0x14), 142
- size of user memory areas (0x13), 141
- SSL ID (0x00), 140
- SSL IDs supported, 139
- start events, OBs (0x82), 148
- station, module status (0x91), 149
- status, communications (0x32), 147
- system areas (0x14), 142
- T memory size (0x14), 142
- types of blocks (0x15), 142
- User memory (0x13), 141
- verify SSL ID (0x00), 140
- version number (0x11), 140
- Start date/time (OB10), 59
- Start event, 72
- Starting a controller name with a hyphen (dash or minus sign), 28
- Starting the controller, 29, 40, 48
- Startup, 42, 52, 54, 101
 - autostart, 45
 - configuring, 52, 54
 - restart method, 49
- Station configuration editor, 14, 16, 19, 20
- Station failure, 138
- STATUS, 70, 80
- Status indicators, 1, 33
- STEP 7, 51
 - accessing WinLC, 51
 - configuring PC station, 14
 - configuring the controller, 51
 - connecting to WinLC, 24, 26
 - hardware configuration, 26
 - OPC server, 111
 - renaming the controller (invalid characters), 28
 - SSL IDs supported, 139
 - system status list (SSL), 139
- STOP, 30, 33
- Submodule, 14, 17, 19, 20, 24
 - CP card, 24

- creating, 20
- System clock, 82
- System function blocks (SFBs), 80
- System functions (SFCs), 75
- System requirements, 5
- System status list, 139
 - 0x00, 140
 - 0x11, 140
 - 0x12, 141
 - 0x13, 141
 - 0x14, 142
 - 0x15, 142
 - 0x19, 143
 - 0x1C, 143
 - 0x21, 144
 - 0x22, 145
 - 0x23, 145
 - 0x24, 146
 - 0x25, 146
 - 0x31, 147
 - 0x32, 147
 - 0x74, 148
 - 0x82, 148
 - 0x90, 148
 - 0x91, 149
 - 0x92, 149
 - 0x95, 150
 - 0xA0, 150
 - 0xB1, 150
 - 0xB3, 150
 - 0xB4, 150
- assignment of interrupts and errors (0x21), 144
- available SSL IDs, 139
- available SSL IDs (0x00), 140
- backup memory (0x13), 141
- block types (0x15), 142
- C memory size (0x14), 142
- communications performance (0x31), 147
- communications status (0x32), 147
- component identification (0x1C), 143
- contents (SSL IDs supported), 139
- CPU characteristics (0x12), 141
- CPU LED status (0x74), 148
- CPU LED status, local only (0x19), 143
- CPU operating mode (0x24), 146
- DB number and size (0x15), 142
- diagnostic buffer (0xA0), 150
- DP master system (0x90), 148
- DP master system, expanded (0x95), 150
- DP module diagnostics (00B1, 00B3, 00B4), 150
- DP module status (0x91), 149
- DP rack/station status (0x92), 149
- error assignment (0x21), 144
- events that start OBs (0x82), 148
- expanded DP master system (0x95), 150
- FB and FC number and size (0x15), 142
- H CPU LED status (0x74), 148
- I memory size (0x14), 142
- identification, module (0x11), 140
- indexes for an SSL ID (0x00), 140
- interrupt and error assignment (0x21), 144
- interrupt status (0x22), 145
- L memory size (0x14), 142
- LED status for local and redundant modules (0x74), 148
- LED status, local only (0x19), 143
- list SSL IDs (0x00), 140
- Load memory (0x13), 141
- local CPU LED status (0x74), 148
- local module LED status (0x19), 143
- M memory size (0x14), 142
- master system (0x90), 148
- maximum number and size of blocks (0x15), 142

- memory area sizes (0x14), 142
- memory areas (0x13), 141
- mode, CPU (0x24), 146
- module diagnostics (00B1, 00B3, 00B4), 150
- module identification (0x11), 140
- module LED status, local only (0x19), 143
- module LED status, redundant CPU (0x74), 148
- module status (0x91), 149
- OB number and size (0x15), 142
- OB priority class (0x23), 145
- OB start events (0x82), 148
- operating mode (0x24), 146
- order number (0x11), 140
- peer-to-peer memory (0x13), 141
- performance, communications (0x31), 147
- PII memory size (0x14), 142
- PIQ memory size (0x14), 142
- priority class (0x23), 145
- process image partitions (0x25), 146
- Q memory size (0x14), 142
- rack/station status (0x92), 149
- redundant CPU LED status (0x74), 148
- SDB number and size (0x15), 142
- SFC51, 139
- shadow memory (0x13), 141
- size of blocks (0x15), 142
- size of the memory areas (0x14), 142
- size of user memory areas (0x13), 141
- SSL ID (0x00), 140
- SSL IDs supported, 139
- start events, OBs (0x82), 148
- station, module status (0x91), 149
- status, communications (0x32), 147
- system areas (0x14), 142
- T memory size (0x14), 142
- types of blocks (0x15), 142
- User memory (0x13), 141

- verify SSL ID (0x00), 140
- version number (0x11), 140

T

- Technical specifications, 129
- Test for jitter, 86, 135
- Threads, 86, 90, 91
- Time (system clock), 82
- Time error, 72
- Time-delay interrupt, 60, 72
- Time-of-day interrupt, 52, 59, 72
- Timing adjustment, 35
- Transition from STOP to RUN, 40, 42, 58
- Troubleshooting, 137
 - defective state, 33
 - error conditions, 33
 - network problems, 137
- Tuning panel, 35, 83, 86, 90, 91, 97, 101, 107
- Tuning performance, 35, 83, 93, 97

U

- Unbuffered startup, 42
- Uninstalling WinAC, 12
- Uninterruptible power supply (UPS), 44
- Unregister, 48
- URCV, 70, 80
- USEND, 70, 80
- User privileges, 6
- USTATUS, 70, 80

V

- Valid characters for instance name, 28
- Virtual rack, 16
- Virtual slot, 16

W

- WAIT, 100
- Warm restart, 49, 54, 59, 72
- Watchdog timer, 56, 61, 97
- WinAC installation, 9, 10, 11
- Windows, 45

- always on top, 45
- priorities, 91
- UPS settings, 44
- Windows user privileges, 6
- WinLC, 14, 20, 24, 51
 - accessing from STEP 7, 51
 - clock correction factor, 52
 - communicating with I/O, 17
 - configuring communications, 24
 - configuring in PC station, 14
 - configuring with STEP 7, 51
 - connecting to STEP 7, 24, 26
 - control panel, 1
 - CP card submodule, 17, 20, 24
 - features, 2, 129
 - priorities, 91
 - process control, 1
 - properties dialog, 20
 - registering for start at PC boot, 48
 - renaming (invalid characters), 28
 - specifications, 129
 - starting, 48
 - unregistering for start at PC boot, 48
- Work memory, 40, 42
 - saving and restoring, 40

Please check any industry that applies to you:

- Automotive
- Chemical
- Electrical Machinery
- Food
- Instrument and Control
- Non-electrical Machinery
- Petrochemical
- Pharmaceutical
- Plastic
- Pulp and Paper
- Textiles
- Transportation
- Other _____

Mail your response to:

Siemens Energy & Automation, Inc.
ATTN: Technical Communications M/S 5518
One Internet Plaza
Johnson City TN USA 37604

Include this information:

From

Name: _____
Job Title: _____
Company Name _____
Street: _____
City and State: _____
Country: _____
Telephone: _____