SIEMENS

SIMATIC

Windows Logic Controller (WinLC)

User Manual

C79000–G7076–C217–03

Edition: 3

Safety Guidelines

This manual contains notices which 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 that death, severe personal injury or substantial property damage **will** result if proper precautions are not taken.

⚠ Warning

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

⚠ Caution

indicates that minor personal injury or property damage can result if proper precautions are not taken.

Note

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

Qualified Personnel

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

Correct Usage

Note the following:

⚠ Warning

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

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

Trademarks

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

Some of other designations used in these documents are also registered trademarks; the owner's rights may be violated if they are used by third parties for their own purposes.

Siemens AG
Automation and Drives (A&D)
Industrial Automation Systems (AS)
Postfach 4848, D- 90327 Nürnberg

© Siemens AG 1999
Technical data subject to change.

Siemens Aktiengesellschaft

Preface

The Windows Logic Controller (WinLC) is a software package that provides the programmable logic controller (PLC) functionality for the Windows Automation Center (WinAC) Basis package. It is fully compatible with the SIMATIC product family. You can use any of the SIMATIC products, such as the Windows Control Center (WinCC) with WinLC.

WinLC communicates over PROFIBUS-DP to control the distributed I/O, such as ET 200M. WinLC can communicate to STEP 7 or other programming software on another computer over PROFIBUS–DP, Ethernet, or MPI networks.

Audience

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

Scope of the Manual

This manual describes the features and the operation of WinLC version 3.0.

Other Manuals

You can find information in the online help for STEP 7 and for the WinLC. For more information, refer to the following manuals:

Title	Content
System Software for S7-300 and S7-400 Program Design Programming Manual	This manual provides basic information on the structure of the operating system and of a user program of the WinLC. Use this manual when creating a user program with the STEP 7 automation software.
S7-300 and S7-400 System and Standard Functions Reference Manual	The 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.
STEP 7 User Manual	This manual explains the main usage and the functions of the STEP 7 automation software. This manual provides you with an overview of the procedures used to configure and program the WinLC.
SIMATIC NET PROFIBUS User Manual	This manual provides information about PROFIBUS–DP communications and setting up PROFIBUS networks.

Additional Assistance

If you have any questions not answered in this or one of the other STEP 7 manuals, if you need information on ordering additional documentation or equipment, or if you need information on training, please contact your Siemens distributor or sales office.

To contact Customer Service for Siemens in North America:

- Telephone:
  - (609) 734–6500
  - (609) 734–3530
- E-mail:
  - ISBU.Hotline@sea.siemens.com
  - simatic.hotline@sea.siemens.com
- Internet:
  - http://www.aut.sea.siemens.com/winac/
  - http://www.aut.sea.siemens.com/simatic/support/index.htm
  - http://www.ad.siemens.de/support/html_76/index.shtml

To contact Customer Service for Siemens in Europe:

• Telephone:	++49 (0) 911 895 7000
• Fax:	++49 (0) 911 895 7001
• E-mail:	simatic.support@nbgm.siemens.de
• Internet:	http://www.ad.siemens.de/simatic–cs

Contents

Figures

Tables

Product Overview 1

The Windows Logic Controller (WinLC) provides process control from your computer. As part of the SIMATIC family of automation products, WinLC is fully compatible with any of the SIMATIC products, such as the STEP 7 programming software and the Windows Control Center (WinCC). The SIMATIC family of automation tools helps to make the WinLC controller a powerful solution for your automation needs.

WinLC can communicate to STEP 7 remotely over PROFIBUS, Ethernet, or MPI networks. WinLC controls distributed I/O, such as ET 200M over PROFIBUS–DP.

1.1 Controlling Your Process with WinLC

WinLC provides a computer-based solution for your automation projects. As shown in Figure 1-1, WinLC connects a PC-based controller over a PROFIBUS, Ethernet, or MPI network to the distributed I/O that connect to the process or automation project. You can also use the following standard SIMATIC products with WinLC:

- STEP 7 automation software allows you to design, download, test, and monitor the user program that runs on WinLC.
- WinCC provides a human-machine interface (HMI) for monitoring your process.

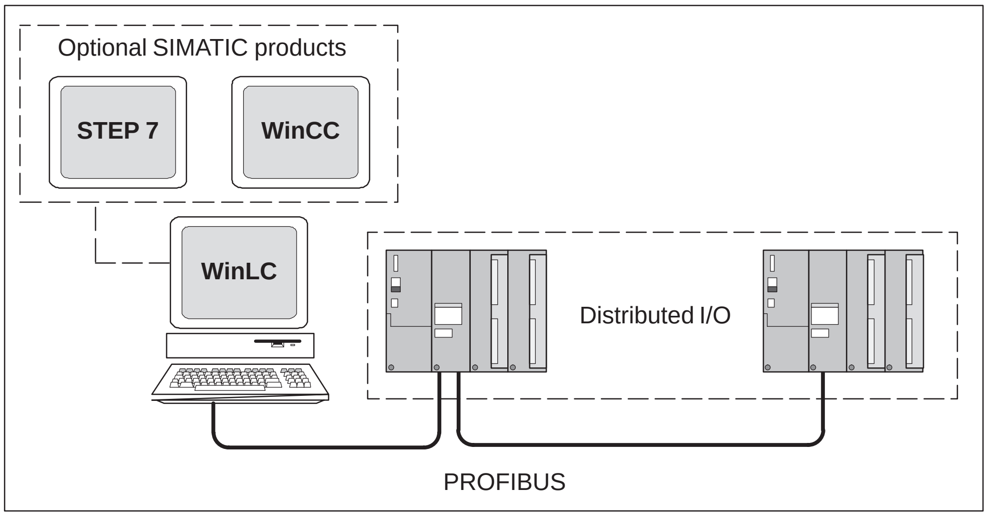


Figure 1-1 Components of WinLC

Operational Features of WinLC

WinLC is a PC-based logic controller in the family of S7 controllers (S7-300 and S7-400). This controller is fully compatible with the automation tools provided by the SIMATIC family of products, such as the STEP 7 programming software and WinCC.

The WinLC controller has four accumulators and supports distributed I/O over a PROFIBUS–DP network. For more information about the operational features of the WinLC, see Chapter 5.

System Requirements

To run WinLC, your computer must meet the following criteria:

- A personal computer (PC) with the following:
  - Pentium processor running at 166 MHz or faster (recommended)
  - 64 Mbytes RAM (recommended)
  - Microsoft Windows NT version 4.0 (or higher), with service pack 3 (or higher) required
- A color monitor, keyboard, and mouse or other pointing device (optional) that are supported by Microsoft Windows NT
- A hard drive with 3 Mbytes of free space
- At least 1 Mbyte free memory capacity on drive C for the Setup program (Setup files are deleted when the installation is complete.)
- An installed CP card connected to a PROFIBUS–DP network for distributed I/O communication.

The product has been tested, and operated successfully, on machines as slow as a 486 processor running at 66 MHz with 24 Mbytes RAM operating on a Windows NT platform. WinLC has also been successfully tested on high-end PCs with dual Pentium processors.

1.2 Additional Features for WinLC Version 3.0

The following features have been added in version 3.0 of WinLC:

- WinLC communicates with STEP 7 and SIMATIC Computing over PROFIBUS, MPI, or Ethernet networks. SIMATIC HMI products are also supported. The recommended version of STEP 7 is STEP7 V5.0 SP3 or higher, but WinLC can be used with earlier versions of STEP 7 (described in section 4.2).
- WinLC is enabled to work with different PROFIBUS cards such as the CP5613. Purchase the CP card separately.
- With STEP7 V5.0 SP3 or higher, WinLC is configured as a PC station.

1.3 Storing the Date (Questions about Year 2000 or “Y2K”)

WinLC stores dates in a two-digit format (for example, 1999 is stored as “99”). WinLC correctly interprets “00” as being 2000. Years are stored from 84 (for 1984) to 83 (for 2083).

1.4 Known Problems

Connecting WinLC to (MPI) Networks

The MPI address of WinLC is independent of the address set by the hardware configuration application, but is determined by the MPI card of the PC. You should always leave the WinLC node address set to MPI =2.

2 Setting Up WinLC Software

Chapter Overview

To use WinLC for process control, you must install and authorize the WinLC software on your computer. Your computer must have a communications processor card (CP card) installed in it. Once the card and software are installed, you must configure WinLC for communication to a distributed I/O network using the CP card. This chapter describes the steps necessary to prepare WinLC for process control.

2.1 Overview of the WinLC Installation

As shown in Figure 2-1, you must install the following components:

- Communications processor (CP) card (purchased separately)
- WinLC software

WinLC is certified to run with the CP5412 using SIMATIC NET DP 5412 V5.1 with Service Pack 1.

The WinLC 3.0 comes with the DP 5412 version 5.2 (no service pack required) and the DP 5613 version 2.0 drivers.

You install these products on your computer and connect WinLC to the distributed I/O over your network.

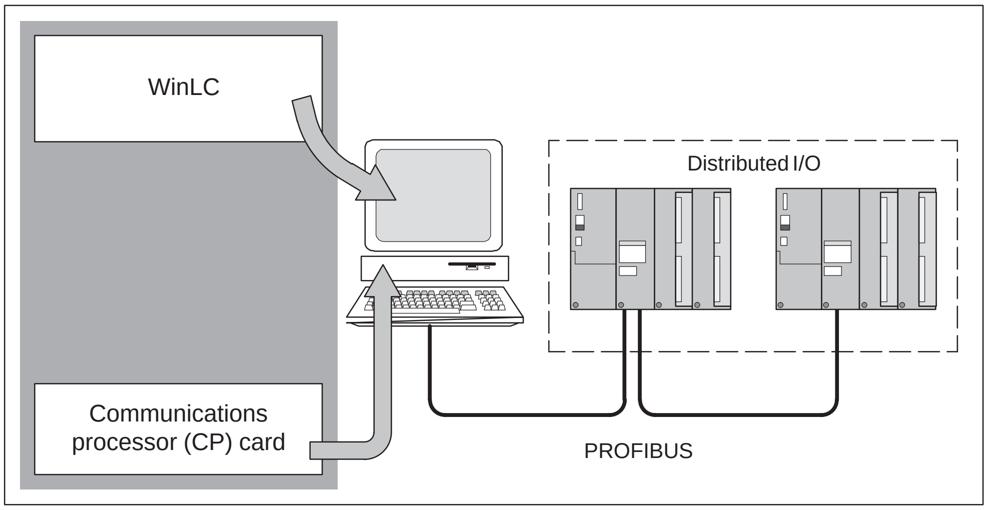


Figure 2-1 Installing the Components for WinLC

Note

The Setup program for WinAC Basis allows you to install WinLC as an NT service.

Each component must be installed separately on your computer. Refer to the documentation for each component for the specific instructions for installing that component. If you are installing the STEP 7 software (or other SIMATIC software package), refer to the installation procedures for that product.

You must perform the following tasks to install the components of WinLC:

- You must install the CP card in your computer. For information about this installation procedure, refer to the documentation for the specific CP.
- Using the Setup program for WinAC Basis, you install the software for WinLC. The DP drivers are included during the installation. See Section 2.2 for information about installing WinLC.
- You must install the authorizations for WinLC. For information about authorizing the WinAC Basis software, see Section 2.4.

After you have installed WinLC and the CP card, you must configure configure the CP for use by WinLC. Then you can connect your computer to the distributed I/O network.

- Chapter 6 provides guidelines for planning the PROFIBUS network. For more information about distributed I/O and PROFIBUS networks, refer to the *SIMATIC NET PROFIBUS User Manual* and to the documentation for the distributed I/O.

2.2 Installing the WinLC Software

WinAC Basis includes a Setup program which executes the installation automatically. The screen prompts guide you step by step through the installation procedure. This Setup program allows you to install any or all of the elements of WinAC Basis. To install only WinLC, deselect the other components of WinAC Basis and select to install only WinLC.

During installation, the 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. If you wish, you can run the authorization program immediately or continue the installation and execute the authorization later.

See Section 2.4. for a description of how to run the authorization program.

Starting the Installation Program

The Setup program guides you step by step through the installation process. You can switch to the next step or to the previous step from any position. To start the installation program, proceed as follows:

1. Start the dialog box for installing software under Windows NT by double-clicking on the **Add/Remove Programs** icon in the Control Panel.
2. Click on “Install...”
3. Insert disk 1 or the CD-ROM and click on “Next.” Windows NT searches automatically for the installation program SETUP.EXE.
4. Follow the instructions displayed by the installation program step by step.
5. When prompted by the software, insert the WinLC authorization diskette in drive A. For more information about authorizing the WinLC software, see Section 2.4.

Once the installation has been completed successfully, a message to that effect is displayed on the screen.

Note

WinLC may be configured to connect to STEP 7 on the same PC as WinLC or on another PC as described in Sections 3.1 and 3.2.

If a Version of WinLC Is Already Installed

If the installation program finds another version of WinLC on the programming device, the program reports this and prompts you to decide how to proceed by offering the following choices:

- Abort the installation so that you can uninstall the old WinLC version under Windows NT and then start the installation again.
- Continue the installation and overwrite the old version with the new version.

Your software is better organized if you uninstall any older versions before installing the new version. Overwriting an old version with a new version has the disadvantage that if you then uninstall, any remaining components of the old version are not removed.

Note

If you are installing WinLC version 3.0 on a computer that has an existing CP card, you must configure WinLC for the correct card. See Chapter 3 for CP card configuration instructions.

Troubleshooting Any Errors That Occur during Installation

The following errors may cause the installation to fail:

- Initialization error immediately after starting Setup: The SETUP.EXE program was probably not started under Windows NT.
- Not enough memory: You need at least 3 Mbytes of free space on your hard disk.
- Bad disk: Verify that the disk is bad, then call your local Siemens representative.

Running WinLC as an NT Service

The Setup program allows you to choose whether to install WinLC as an NT service. You must have administrative privileges to install WinLC as a service. By running as an NT service, WinLC starts automatically any time you start the computer. You can use the Windows NT control panel to change the configuration of WinLC to be started manually instead of automatically.

You can use the CPU panel to register and unregister WinLC as an NT service. See Section 4.7.

To change the behavior of WinLC from starting automatically to starting only when manually executed (either from the **Start** menu or by double-clicking on the WinLC icon), use the Windows NT control panel:

1. From the **Start** menu, select **Control Panel**.
2. Double-click on the **Services** icon to open the “Services” dialog box.
3. Select the entry for “SIMATIC WinLC”. Notice that the Startup behavior is listed as “Automatic”.
4. Click on the “Startup” button to display the dialog box.
5. Select “Manual” and click on the “OK” button. Notice that the Startup behavior is now listed as “Manual”.
6. Close the “Services” dialog box.

After changing WinLC to start manually, you must open the Services dialog box and use the “Start” or “Stop” buttons every time you want to start or stop WinLC.

2.3 Uninstalling the WinLC Software

Use the usual Windows NT procedure to uninstall the WinLC software:

1. Start the dialog box for installing software under Windows NT by double-clicking on the **Add/Remove Programs** icon in the Control Panel.
2. Select the WinLC entry in the displayed list of installed software. Click on the “Add/Remove...” button to uninstall the software.
3. If the “Remove Shared Components” dialog box appears, click the “No” button if you are unsure how to respond.

Caution

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

The Readme file on the authorization diskette contains guidelines for installing, transferring, and removing the authorization for WinLC. If you do not follow these guidelines, the authorization for WinLC may be irretrievably lost. Losing the authorization would prohibit you from modifying any program that was downloaded to WinLC and from downloading another program to WinLC.

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

2.4 Authorization

WinAC Basis requires a product-specific authorization (or license for use). The software is therefore copy-protected and can be used only if the relevant authorization for the program or software package is found on the hard disk of the computer.

Note

If you remove the authorization, the WinLC controller continues to operate; however, you cannot modify the program being executed and you cannot download a new program. You are still permitted to change from RUN mode to STOP mode, and the controller continues to execute the user program. You are still permitted to create and reload archive files.

A notification message appears every six minutes to alert you that the authorization is missing.

If you install an authorization while the WinLC controller is running, you must also change the operating mode of the controller before the authorization takes effect.

Authorization Disk

An authorization diskette is included with the software. It contains the authorization and the program (AUTHORSW) required to display, install, and remove the authorization.

There are separate authorization diskettes for each of the SIMATIC automation software products. You must install the authorization for each product as part of the installation procedure for that software.

Caution

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

The Readme file on the authorization diskette contains guidelines for installing, transferring, and removing the authorization for WinLC. If you do not follow these guidelines, the authorization for WinLC may be irretrievably lost. Losing the authorization would prohibit you from modifying any program that was downloaded to WinLC and from downloading another program to the WinLC.

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

When you install your software for the first time, a message prompts you to install the authorization. To install the authorization for WinLC, follow these steps:

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

The authorization is transferred to the hard drive (C), and your computer registers the fact that the authorization has been installed.

Note

Always enter drive C as the destination drive for the authorization for WinLC.

If you attempt to start WinLC and there is no authorization available for the software, a message informs you of this. If you want to install the authorization, use the AUTHORSW program on the authorization diskette. This program allows you to display, install, and remove authorizations.

Removing an Authorization

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

To transfer the authorization back to the authorization diskette, follow these steps:

1. Insert the original authorization diskette in your floppy disk drive.
2. Start the program AUTHORSW.EXE from the authorization diskette.
3. From the list of all authorizations on drive C, select the authorization to be removed.
4. Select the menu command **Authorization ▸ Transfer...**.
5. In the dialog box, enter the target floppy drive to which the authorization will be transferred and confirm the dialog box.
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. You must use the authorization diskette to remove any existing authorizations.

If a fault occurs on your hard disk before you can back up the authorization, contact your local Siemens representative.

2.5 Troubleshooting the Installation

Installing Computing on a Computer That Is Running WinLC

Do not install the SIMATIC Computing software on a computer that is actively running WinLC during the installation of the Computing software. Since these two products use the common resources, this could cause the files to become corrupted. Always stop the execution of WinLC before installing the Computing software.

Caution

Do not install any component of WinAC (such as WinLC) on a computer while any other component of WinAC (such as WinLC, the SIMATIC Computing SoftContainer, programs that use the SIMATIC controls provided by Computing, or the panel for the CPU 416–2 DP ISA, or other slot PLC) is being executed (are currently running) on that computer.

Since SIMATIC Computing, WinLC, and other elements of WinAC use common files, attempting to install any component of the WinAC software when any of the components of WinAC are being executed by the computer can corrupt the software files. Always ensure that the following programs are not running when you install WinLC:

- WinLC
- Panel for CPU 416–2 DP ISA, or any slot PLC
- SIMATIC Computing SoftContainer
- TagFile Configurator
- Tool Manager
- SIMATIC Computing OPC Configuration
- SIMATIC Computing Configuration
- Any program (such as a program created in Visual Basic) that uses one of the SIMATIC controls provided by Computing

Uninstalling (Removing) the Computing Software While WinLC Is Running

If WinLC is being executed during the uninstallation procedure for the Computing software, WinLC experiences a connection error and loses connection to the machine or process. To recover from the connection error, follow these steps::

1. Using the Windows NT Task Manager, end the process for WinLC (S7wlcapx.exe).
2. If the WinLC panel is open, close the panel.
3. Restart WinLC to reconnect to the machine or process.

Warning

Uninstalling (removing) the Computing software at the same time that WinLC is being executed on that computer causes WinLC to be disconnected from the machine or process that it is controlling. This could cause unpredictable process operation, which could result in death or serious injury to personnel, and/or damage to equipment.

If you cause WinLC to lose connection to the process by uninstalling Computing, use the Windows NT Task Manager to end the WinLC process (S7wlcapx.exe). If the WinLC panel is open, close the panel. To reconnect WinLC to the machine or process, restart WinLC.

Before removing the Computing software, always ensure that the WinLC controller has been shut down and that the WinLC software is not being executed. This helps to ensure that you do not cause WinLC to become disconnected from the machine or process, which could cause process equipment to operate erratically. Always install a physical emergency stop circuit for your machine or process.

Correcting WinLC Connection

There is an important system file created and registered during setup called the Active File. If for any reason the Active File path is not found in the registry, the WinLC controller cannot be started from the view. To correct the problem, perform the following:

- Locate the executable for the WinLC controller (S7wlcapx.exe) and double-click on its icon. The WinLC controller will register the Active File path for you and then start with an empty CPU in STOP mode. To connect to the controller, double-click the view icon. (Note: if you use this method you will not be able to shut down the WinLC controller from the application. You will have to log out of the Windows NT session in order to shut down the WinLC controller.)

The Autostart CPU operation requires that you log in to Windows NT with administrator privileges. It is accessed from the CPU menu of the WinLC panel.

If You Have Not Installed Windows NT 4.0 Service Pack 3

If you have not installed Microsoft Windows NT 4.0 Service Pack 3, you might experience the following problems:

- Failure of CPU indicators (LEDs) to register a change of operating mode
- CPU disconnect errors
- Stack fault messages from the DP authorization software

Microsoft Windows NT 4.0 Service Pack 3 is available as a free download from Microsoft (www.microsoft.com).

Connecting WinLC to (MPI) Networks

The MPI address of WinLC is independent of the address set by the hardware configuration application, but is determined by the MPI card of the PC. You should always leave the WinLC node address set to MPI =2.

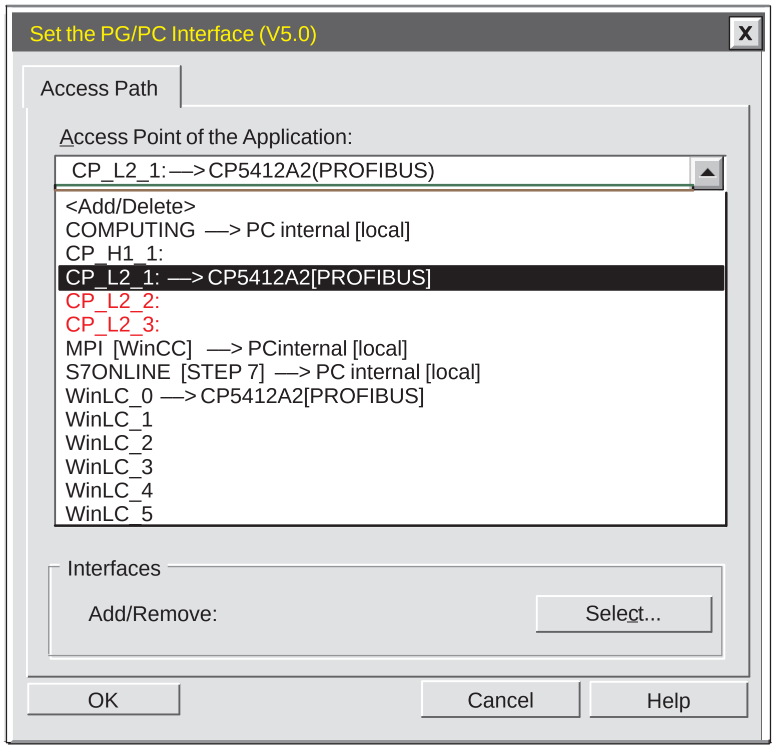
2.6 Setting the Interface for the DP card

You must configure the DP card for communications with the distributed I/O using the installed DP driver.

From WinLC, select **CPU ▸ Setting the PG-PC Interface** to configure the DP card used for I/O communications.

Use the following procedure to configure the communication parameters for the DP card:

1. Using the drop-down list box for the "Access point of application" field, select the CP_L2_1 as the access point. See Figure 2-2.



Access Point of the Application:

CP_L2_1:—>CP5412A2(PROFIBUS)

```
<Add/Delete>
COMPUTING  —> PC internal [local]
CP_H1_1:
CP_L2_1: —> CP5412A2[PROFIBUS]
CP_L2_2:
CP_L2_3:
MPI  [WinCC]   —> PCinternal [local]
S7ONLINE  [STEP 7]  —> PC internal [local]
WinLC_0 —> CP5412A2[PROFIBUS]
WinLC_1
WinLC_2
WinLC_3
WinLC_4
WinLC_5
```

Add/Remove:

Figure 2-2 Setting the Access Point of Application

2.7 Procedure for Configuring WinLC and the DP Driver

WinAC Basis provides software for the WinLC and the DP drivers.

The setup program for WinAC Basis version 3.0 installs the correct driver for the DP card. (The CD for WinAC Basis version 3.0 includes the DP drivers and WinLC V3.0.)

From the WinAC Basis setup program you should select the DP driver that you want to install. Previous installations of DP 5412 A2 V5.1 (plus service pack) will work with this release. However, it is strongly recommended that you use the version included with this release.

Installing the Hardware and Software Elements of WinLC

You must install the DP card and the WinLC software.

1. Install the DP card in your computer. For more information about installing the DP card, refer to the documentation that ships with the card.

Note

The default settings of the DP card should be correct for use in your computer. If the addressing of the DP card conflicts with another card in your computer, you can change the DIP switch settings for the DP card. Refer to the Product Information document that ships with the DP card .

2. Install the WinLC software. For more information about installing the WinLC software, see Section 2.2.

Checking the Connection to the CP_L2_1.ldb Database File

Use the following procedure to test whether the configuration of the DP database has been successful:

1. Start WinLC. For more information about starting WinLC, see Section 4.7.
   – If WinLC is running as an NT service: use the “Services” dialog box to start WinLC. Select the **Start ▸ Settings ▸ Control Panel** menu command to open the Windows NT control panel, then click on the “Services” icon.
   – If WinLC is not running as an NT service: select the **CPU ▸ Start WinLC Controller** menu command from the CPU panel to start the WinLC controller.
2. From the CPU panel, click on the “RUN” button or select the **CPU ▸ RUN** menu command to place WinLC in RUN mode.

If no error messages display, the components of WinLC were installed and configured correctly.

Connecting SIMATIC Client Software to WinLC 3

Chapter Overview

WinLC allows you to connect to SIMATIC products, such as STEP 7, WinCC, and ProTool Pro across networks using MPI, PROFIBUS, or H1 communications networks. Refer to the product manuals for specific setup information. Examples of STEP 7 setup are provided in this chapter. Redirecting communications from MPI as done in earlier versions of WinLC is no longer supported or available.

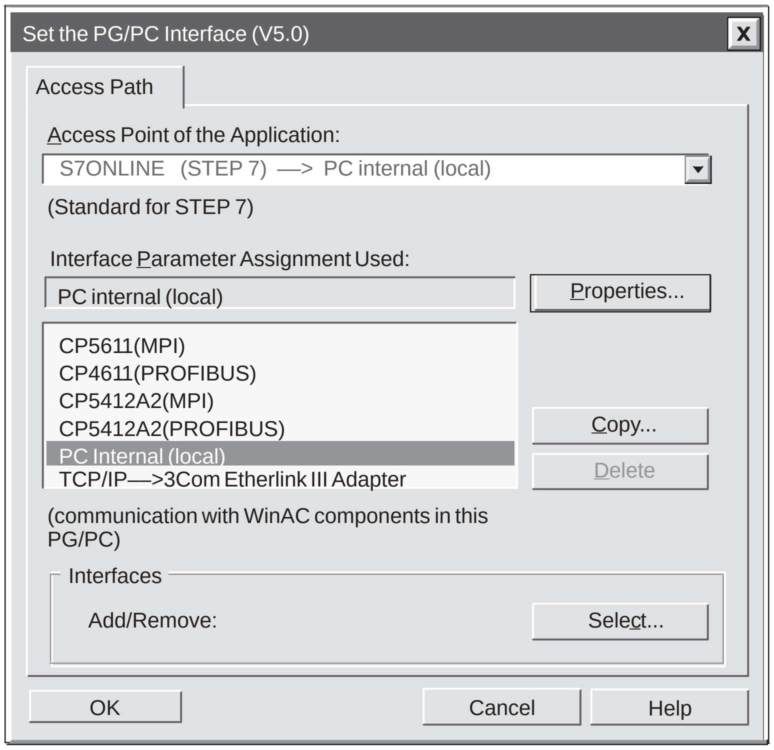
Section	Description	Page
3.1	Connecting STEP 7 to WinLC on the Same Computer	3-2
3.2	Connecting STEP 7 to WinLC on a Different Computer	3-3
3.3	Connecting STEP 7 to Hardware PLCs	3-7

3.1 Connecting STEP 7 to WinLC on the Same Computer

Use the following procedure to configure STEP 7 for communicating with WinLC on the same computer:

1. From WinLC open the interface tool as follows: (**CPU ▸ Setting the PG/PC Interface**)
2. Use these steps to configure STEP 7 as the local access point:
   – From the "Access Point of the Application" drop-down list, select **S7ONLINE (STEP 7)** (Figure 3-1).
   – From the "Interface Parameter set used" drop-down list, select **PC Internal (local)** for the interface parameter.

STEP 7 is now configured to communicate with WinLC on this computer.


Set the PG/PC Interface (V5.0)

Access Path

Access Point of the Application:
S7ONLINE (STEP 7) —> PC internal (local)
(Standard for STEP 7)

Interface Parameter Assignment Used:
PC internal (local)

CP5611(MPI)
CP4611(PROFIBUS)
CP5412A2(MPI)
CP5412A2(PROFIBUS)
PC Internal (local)
TCP/IP—>3Com Etherlink III Adapter

(communication with WinAC components in this PG/PC)

Interfaces
Add/Remove:

Figure 3-1 Setting the PG/PC Interface for PC Internal (local)

3.2 Connecting STEP 7 to WinLC on a Different Computer

As shown in Figure 3-2, you can connect STEP 7 on one computer to a WinLC on a different computer. You must define the network connection over which STEP 7 and WinLC communicate by setting the PG/PC interface on the remote computer.

The remote computer must have STEP 7 installed, and the computer to which you wish to connect must have WinLC installed.

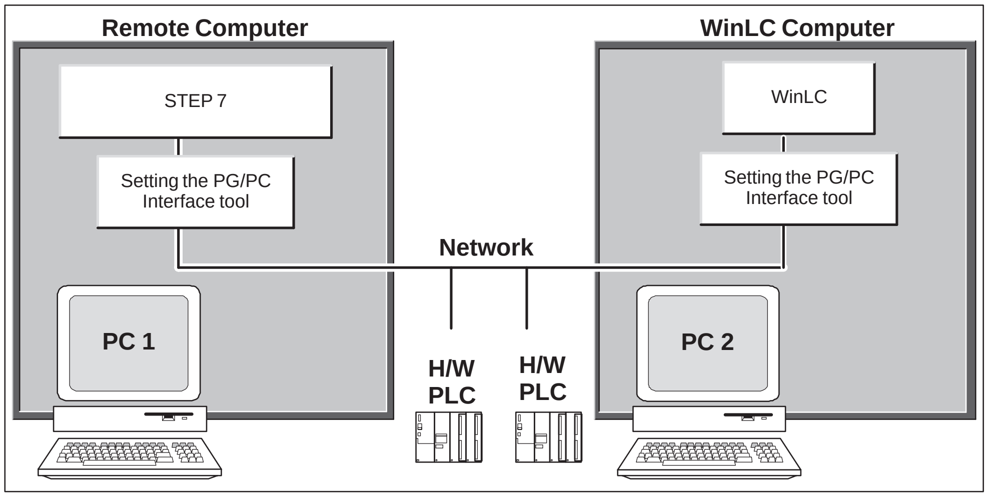
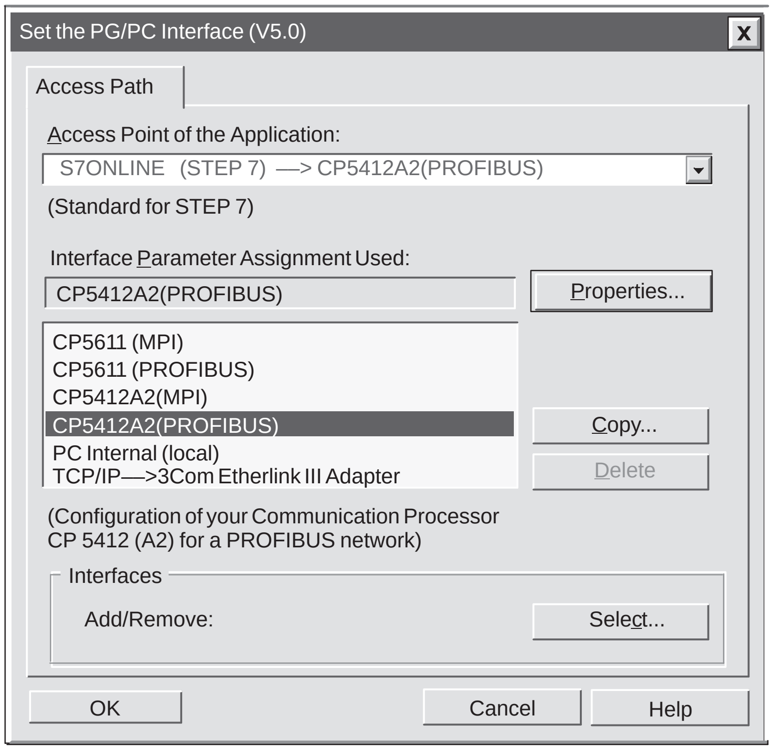


Figure 3-2 Connecting STEP 7 to WinLC over a network

From the computer on which STEP 7 resides, follow these steps to configure STEP 7 for communicating with WinLC on a remote computer:

1. Access the interface configuration tool from SIMATIC Manager through **Options ▸ Set the PG/PC Interface**.



Access Point of the Application:
S7ONLINE (STEP 7) —> CP5412A2(PROFIBUS)
(Standard for STEP 7)

Interface Parameter Assignment Used:
CP5412A2(PROFIBUS)

CP5611 (MPI)
CP5611 (PROFIBUS)
CP5412A2(MPI)
CP5412A2(PROFIBUS)
PC Internal (local)
TCP/IP—>3Com Etherlink III Adapter

(Configuration of your Communication Processor CP 5412 (A2) for a PROFIBUS network)

Add/Remove:

Figure 3-3 Setting the PG/PC Interface for CP5412A2(PROFIBUS)

2. From the “Access Point of Application” drop-down list, select **S7ONLINE (STEP7).**
3. Select the interface parameter from the parameter set that corresponds to your network communications path.
   – For MPI communication, select the MPI interface, for example, **CP5611 (MPI).**
   – For PROFIBUS–DP communication, select the PROFIBUS–DP interface, for example, **CP5412A2(PROFIBUS).**

     The WinLC’s PROFIBUS card must be properly configured through **Setting the PG/PC Interface** before WinLC is visible to other PGs on the PROFIBUS–DP network **(S7ONLINE (STEP7) ––> Profibus...**. It must be set for “PG is the only master on the bus.”
   – For H1 communication, select the TCP/IP interface, for example, **TCP/IP –> 3Com Etherlink III Ada...** You must have the NCM Options package for H1 communication and STEP 7 V5 SP3.

Note

NetPro cannot reconfigure the MPI or H1 addresses or the bus parameters of a WinLC from a different computer. The required CP cards are not controlled by WinLC. This can only be done via the local Setting the PG/PC Interface application. The PROFIBUS node address and bus parameters can be reconfigured remotely. The WinLC is the master of its own PROFIBUS I/O card.

From the computer on which WinLC resides, the communication path(s) to networks with computer(s) running STEP 7 must be configured. Ten access points are installed by WinLC. Each access point can point to one of the installed interfaces.

Example:

WinLC_0 —> none

WinLC_1 —> CP5613_5614(PROFIBUS)

WinLC_2 —> none

WinLC_3 —> none

WinLC_4 —> none

WinLC_5 —> none

WinLC_6 —> CP5611 (MPI)

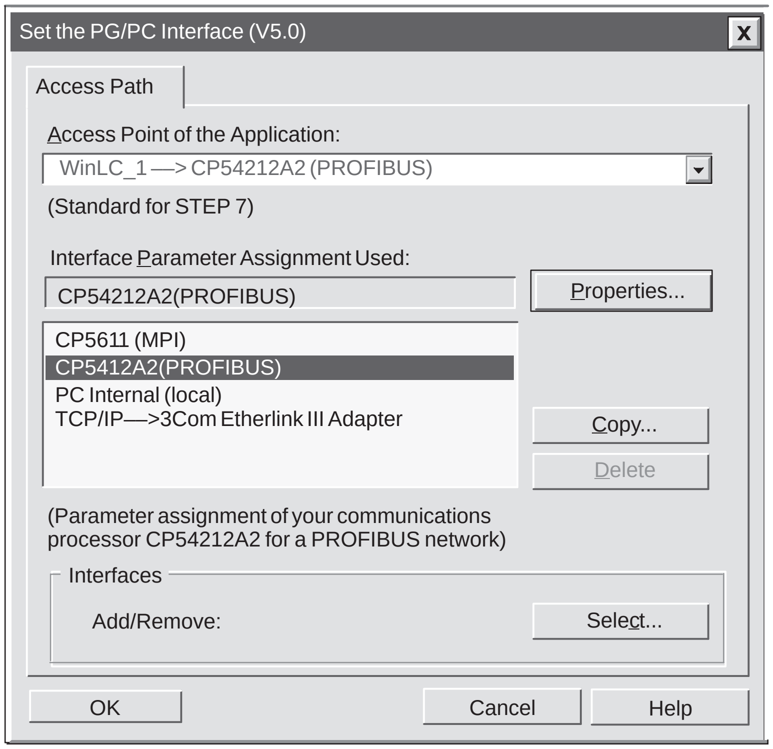
WinLC_7 —> none

WinLC_8 —> none

In this example, the WinLC 3.0 can be accessed through two cards at the same time. The WinLC cannot be reached through cards that are not assigned to an access point.

To configure one of the access points, follow these steps:

1. Access the interface configuration tool through WinLC. Use (**CPU ▶ Setting the PG/PC Interface**.



Set the PG/PC Interface (V5.0)

Access Path

Access Point of the Application:

WinLC_1 —> CP54212A2 (PROFIBUS)

(Standard for STEP 7)

Interface Parameter Assignment Used:

CP54212A2(PROFIBUS)

CP5611 (MPI)
CP5412A2(PROFIBUS)
PC Internal (local)
TCP/IP—>3Com Etherlink III Adapter

(Parameter assignment of your communications processor CP54212A2 for a PROFIBUS network)

Interfaces

Add/Remove:

Figure 3-4 Setting the PG/PC Interface for the CP Card

2. From the “Access point of application” drop-down list, select **WinLC_1.**
3. Select the interface parameter from the parameter set that corresponds to your network communications path, for example **CP5412A2 (PROFIBUS**).

Repeat steps 2 and 3 as needed to configure each access point used to communicate to a network.

3.3 Connecting STEP 7 to Hardware PLCs

Follow the procedure above for **Connecting STEP 7 to WinLC on a Different Computer.** STEP 7 has now been configured to communicate with WinLC on the remote computer as well as hardware PLCs on that network. You can use any of the STEP 7 tools or functionality across the network. Figure 3-2 shows how STEP 7 is connected to the configured network.

Note

The cyclic distribution of PROFIBUS bus parameters cannot be performed by WinLC.

Running the WinLC Software 4

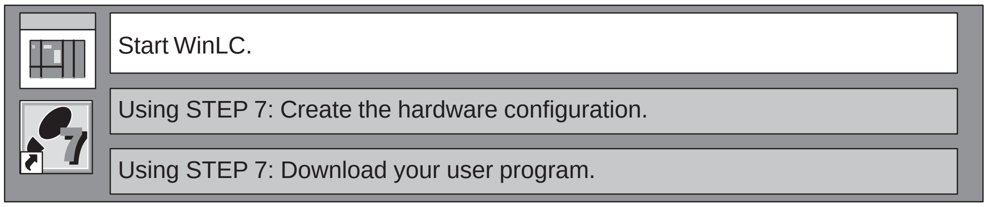
Chapter Overview

WinLC provides a CPU panel that allows you to control the operation of the WinLC controller. With this panel, you can perform the following tasks:

- Monitor the status
- Change the operating mode
- Register and unregister WinLC as an NT service
- Perform a cold restart or a warm restart
- Enable the Autostart feature of WinLC
- Monitor the scan cycle
- Tune the operations of WinLC
- Change the language for the WinAC applications
- Create levels of access and security for WinLC
- Change the password for WinLC

4.1 Starting the WinLC Software

Figure 4-1 provides an overview of the steps required for configuring the hardware and downloading the user program to WinLC.



Start WinLC.

Using STEP 7: Create the hardware configuration.

Using STEP 7: Download your user program.

Figure 4-1 Starting WinLC

Getting Started

Use the following procedure to start WinLC:

1. Go to the main Windows NT taskbar and click on the “Start” button.
2. Select the WinLC software from the “Start” menu (**Start** ▸ **SIMATIC** ▸ **PC Based Control** ▸ **Windows Logic Controller**).

You can change the operating mode of the WinLC controller from STOP to RUN by clicking on the RUN or RUN-P button of the CPU panel. When you change the operating mode, the status indicators on the panel also change. For more information about using the CPU panel, see Section 4.4 or Section 5.1.

WinLC opens with a CPU panel, as shown in Figure 4-2.

Setting Network Connections for STEP 7

After you install the WinLC software, STEP 7 communication is configured to be directed to WinLC instead of to hardware PLCs. You can use the CPU panel of WinLC to configure communication to go to either hardware PLCs or WinLC.

From the WInLC panel, select **CPU** ▸ **Setting the PG/PC Interface** menu. Appendix 3 describes how to configure network settings.

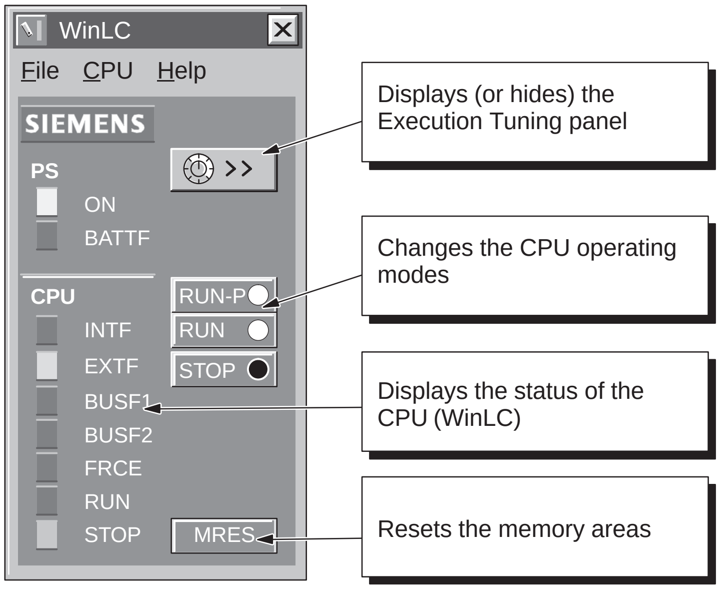

WinLC
File CPU Help
SIEMENS
PS
ON
BATTF
CPU
INTF
EXTF
BUSF1
BUSF2
FRCE
RUN
STOP
RUN-P
RUN
STOP
MRES
Displays (or hides) the Execution Tuning panel
Changes the CPU operating modes
Displays the status of the CPU (WinLC)
Resets the memory areas

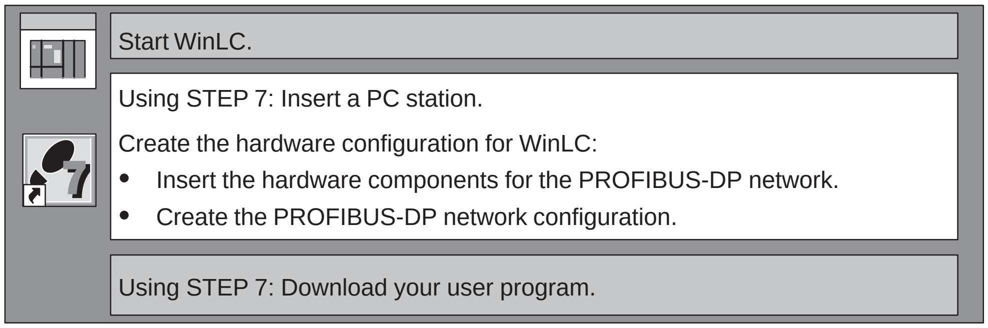
Figure 4-2 CPU Panel for WinLC

4.2 Creating the Hardware Configuration

The hardware configuration defines the network addresses and the distributed I/O (DP) for the WinLC controller. It also defines the default operating parameters, such as the minimum scan cycle time. As shown in Figure 4-3, you must use the STEP 7 programming software to configure WinLC:

- Use the SIMATIC Manager to create a project and a PC station.
- Use the Hardware Configuration to configure WinLC and the distributed I/O.

For information about using the STEP 7 programming software, refer to the *STEP 7 User Manual* or to the online help for the STEP 7 software.



Start WinLC.

Using STEP 7: Insert a PC station.

Create the hardware configuration for WinLC:
- Insert the hardware components for the PROFIBUS-DP network.
- Create the PROFIBUS-DP network configuration.

Using STEP 7: Download your user program.

Figure 4-3 Using STEP 7 to Configure the WinLC

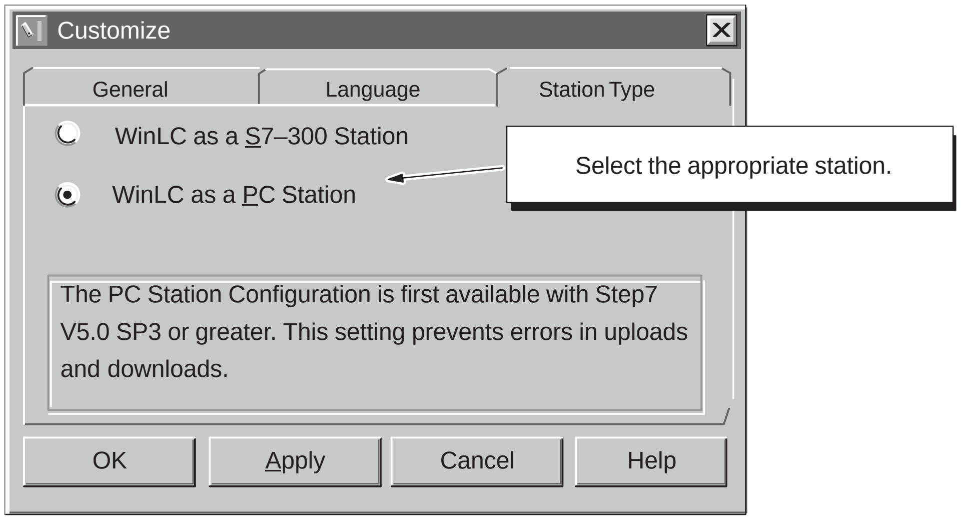
Inserting a Station for WinLC in STEP 7

Before you can create the hardware configuration for WinLC, you must insert a station under your project. For STEP 7 Version 5, Service Pack 3, insert a PC station. STEP 7 V5 SP3 models WinLC as a component in a PC station. (For versions of STEP 7 prior to Version 5, Service Pack 3, you must insert a SIMATIC 300 station.) Use the following procedure for inserting a station:

1. Select (click on) the project (ZEn01_09_STEP7__Zebra).
2. Select the **Insert ▸ PC Station** menu command to insert a station under the project. (To insert a SIMATIC 300 station, select **Insert ▸ SIMATIC 300 Station**.)
3. Select (click on) the station to display the hardware icon for the station.

Note

Certain System Data Blocks have a different structure depending on whether the WinLC is configured in a 300 Station or a PC Station. You must inform WinLC which station type to use via the Panel menu. In WInLC, select **CPU ▸ Options ▸ Customize** and then select the **Station Type** tab**.** Select the appropriate station type as shown in Figure 4-4. Failure to configure the correct station type will cause upload/download errors.



WinLC as a S7–300 Station

[✓ WinLC as a PC Station]

The PC Station Configuration is first available with Step7 V5.0 SP3 or greater. This setting prevents errors in uploads and downloads.

Figure 4-4 Setting the Station Type

Inserting the Hardware Components

You use the hardware configuration of the STEP 7 programming software to configure WinLC:

Note

You cannot use the MPI node of WinLC version 3.0 to configure hardware. WinLC has no effect on any installed MPI card. The MPI address of WinLC should be left at node address 2.

---

1. Select the PC station. If you have STEP 7 without the service pack, select SIMATIC 300 station.
2. Double-click on the Hardware object to open the configuration tool of the STEP 7 software (Figure 4-5).

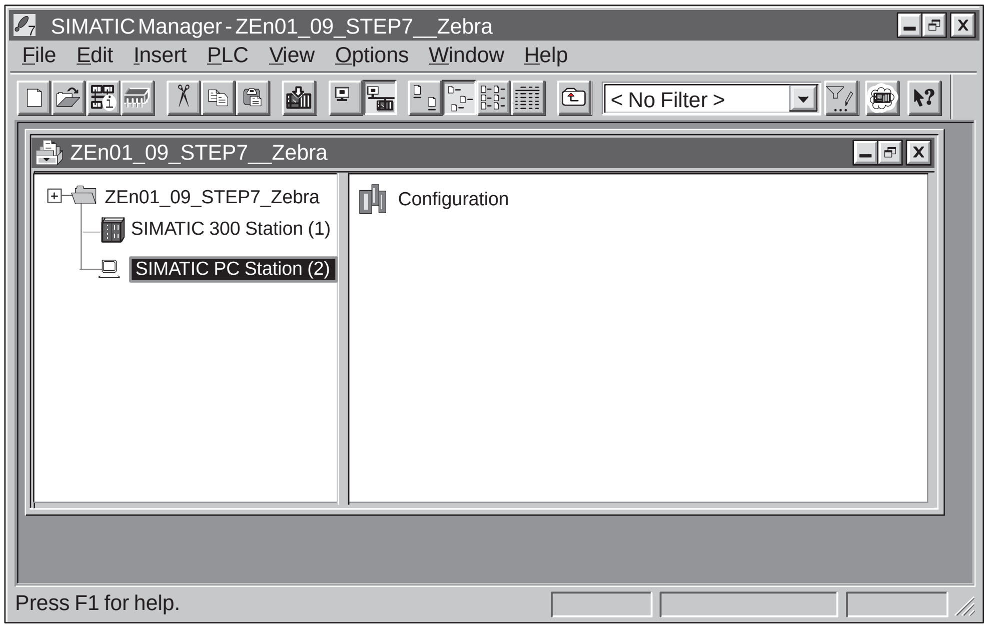


Figure 4-5 Configuring the PC Station in the Sample Project

3. For a PC station:
   - Double-click on the Configuration icon to open the hardware catalog.
   - Select the 2nd slot in the PC display.
   - Select **SIMATIC PC Station ▸ Controller** from the catalog. Click on the WinLC icon.
   - Use the mouse to drag the “WinLC” object into slot 2 of the PC display

Note

With STEP7 V5.0 SP3 or higher, WinLC is configured as a PC station. WinLC 3.0 has features that can only be used if configured in a PC station. In earlier versions of STEP 7 without the service pack, you must use an S7300 station and configure WinLC as version 2.0. See step 4. below.

4. For a SIMATIC 300 station in a version of STEP 7 prior to V5 SP3:
   - Select the **Insert ▸ Hardware Components** menu command to open the hardware catalog.
   - Select and open the “SIMATIC PC Based Control 300/400 ‘” object.
   - Double-click the “WinLC” object. If you do not have STEP 7 V5 SP3, be sure to select WinLC V2.0.
5. In the “Properties – PROFIBUS Node DP Master” dialog box, click on the “New” button to open the “Properties – New Subnet PROFIBUS” dialog box and enter a PROFIBUS subnet or click OK to accept the default of PROFIBUS(1).
6. Click on the “OK” button to enter the default parameters for a PROFIBUS subnet.
7. Select the PROFIBUS(1) subnet.
8. Click on the “OK” button to enter the default subnet and address and to close the “Properties – PROFIBUS Node DP Master” dialog box. WinLC V.3.0 appears as the module in slot 2 of the rack.
9. Select the **Station ▸ Save and Compile** menu command to create the sample hardware configuration for WinLC.

STEP 7 generates the SDBs for the hardware configuration. Exit the Hardware Configuration tool.

4.3 Downloading Your User Program

You use the STEP 7 programming software to download your user program to WinLC (Figure 4-6). See Chapter 3 for information on connecting STEP 7 to WinLC.

Due to limitations of the Microsoft “structured document,” programs downloaded to WinLC are limited to 2500 blocks. This limitation will be addressed in a future release of WinLC.

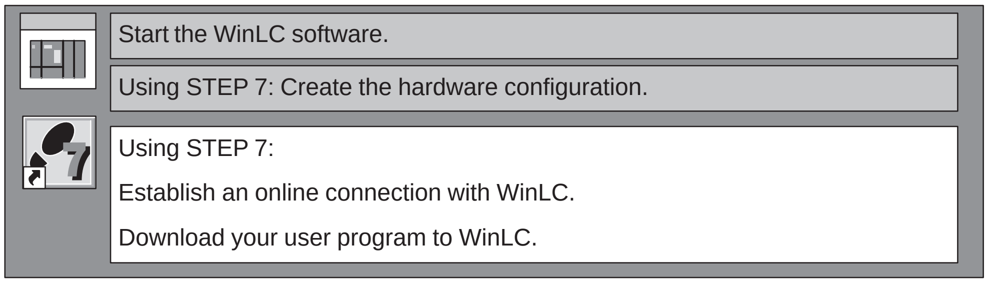

Start the WinLC software.
Using STEP 7: Create the hardware configuration.
Using STEP 7:
Establish an online connection with WinLC.
Download your user program to WinLC.

Figure 4-6 Using STEP 7 to Download the User Program

Accessing WinLC from STEP 7

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

1. Using the SIMATIC Manager, activate the required project window.
2. Select the **View ▸ Online** menu command to change to the “Standard Hierarchy, Online” view.

STEP 7 establishes an online connection to WinLC.

Downloading a User Program from STEP 7

After you have established an online connection to WinLC, you can download your user program:

1. Open the icon for your user program and select the “Blocks” object.
2. Select the **PLC ▸ Download** menu command or click on the Download button.

STEP 7 downloads all of the blocks of your user program, including the System Data Blocks (SDBs), to the WinLC. You can also download individual blocks of the user program.

For more information about downloading programs, see the *STEP 7 User Manual* or the online help for the STEP 7 programming software.

4.4 Executing the User Program

After you have downloaded your user program to WinLC, you use the CPU panel to control the operation of the controller. The CPU panel corresponds to the front panel of an S7 CPU module.

Running the WinLC Software without Valid Authorization

If you lose the authorization for the software, the WinLC controller will continue to run, although with less functionality. You will still be permitted to change from STOP mode to RUN mode, but the controller will not go into RUN-P mode. The controller continues to execute the user program, but you cannot modify the user program, and you cannot download a new program or new blocks to the controller. You can still create and reload archive files.

Using the CPU Panel

The WinLC software starts with a CPU panel. See Figure 4-7. The CPU panel contains the following elements:

- A button for displaying or hiding the tuning panel for adjusting the operation of WinLC (see Section 4.6)
- Three buttons for changing the operating mode of the controller
- Status indicators
- A button for resetting the memory areas

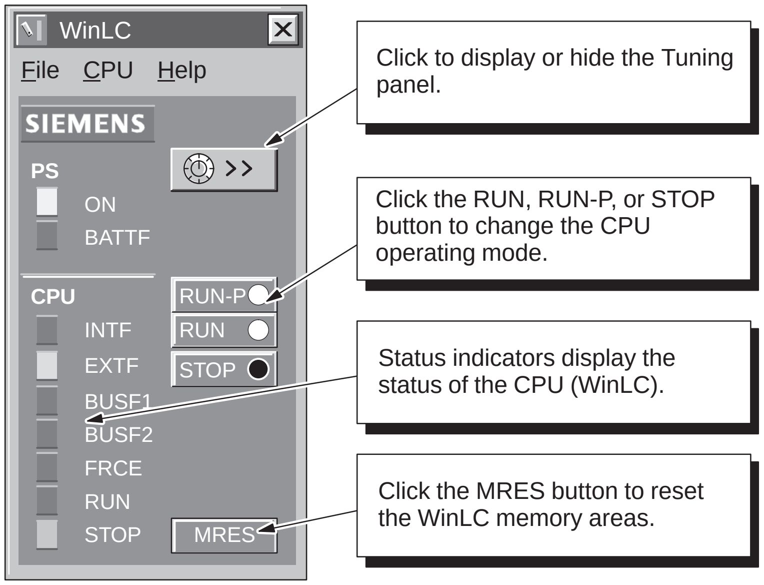

WinLC
File CPU Help
SIEMENS
PS
ON
BATTF
CPU
INTF
EXTF
BUSF1
BUSF2
FRCE
RUN
STOP
>>
RUN-P
RUN
STOP
MRES
Click to display or hide the Tuning panel.
Click the RUN, RUN-P, or STOP button to change the CPU operating mode.
Status indicators display the status of the CPU (WinLC).
Click the MRES button to reset the WinLC memory areas.

Figure 4-7 Using the CPU Panel of WinLC

Selecting the Operating Mode

The RUN, RUN-P, and STOP buttons on the CPU panel correspond to the different operating modes of the controller:

- In STOP mode, the controller is not executing the user program. To download a program that includes SDBs, you must place WinLC in STOP mode. On the transition to STOP mode, the outputs go to a safe state (as configured with the STEP 7 programming software).
- In RUN mode, the controller executes the user program. You cannot download any new user program or logic blocks when the controller is in RUN mode. You can use the STEP 7 programming software to monitor (but not to modify) the variables.
- In RUN-P mode, the controller executes the user program. You can download new programs or logic blocks, and you can use the STEP 7 programming software to modify the variables for testing and debugging.

Clicking on the button places the controller into the selected operating mode. The status indicators on the CPU panel show whether the controller is in RUN mode or in STOP mode.

Selecting a Warm Restart or a Cold Restart

The hardware configuration downloaded with your user program determines the default startup mode for WinLC. (For more information about the restart options, see Section 5.5.) When changing the operating mode of WinLC from STOP mode to RUN mode, you can selectively change the type of restart:

- When you use the **CPU** menu commands (**CPU ▸ RUN** or **CPU ▸ RUN-P**) to change the operating mode, WinLC displays the “Restart Method” dialog box that allows you to select a cold restart or a warm restart. See Figure 4-8. Select the type of restart and click on the “OK” button.
- Using the left mouse button to click on the “RUN” or “RUN-P” buttons on the CPU panel performs a warm restart and does not display the “Restart Method” dialog box.
- As shown in Figure 4-9, using the right mouse button to click on the “RUN” or “RUN-P” buttons displays the “Restart Method” dialog box that allows you to select a cold restart or a warm restart.

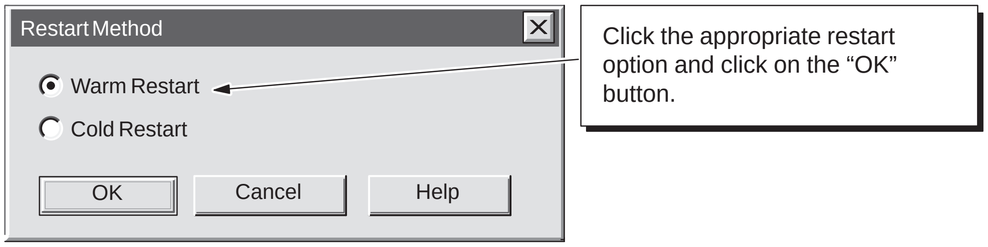
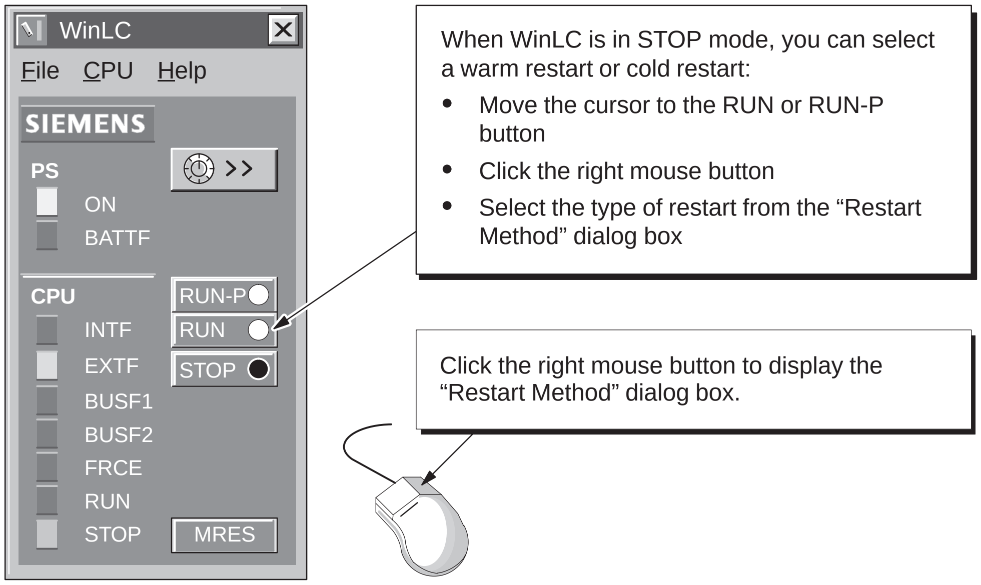

Restart Method
Warm Restart
Cold Restart
OK
Cancel
Help
Click the appropriate restart option and click on the “OK” button.

Figure 4-8 Selecting a Cold Restart or a Warm Restart



When WinLC is in STOP mode, you can select a warm restart or cold restart:

- Move the cursor to the RUN or RUN-P button
- Click the right mouse button
- Select the type of restart from the “Restart Method” dialog box

Click the right mouse button to display the “Restart Method” dialog box.

Figure 4-9 Using the Right Mouse Button to Select the Restart Method

Using the MRES Button to Reset the Memory Areas

The CPU panel provides a MRES button for resetting the memory areas to the default values and deleting the user program. Clicking the MRES button places the controller into STOP mode and performs the following tasks:

- The controller deletes the entire user program, including data blocks (DBs) and system data blocks (SDBs).
- The controller resets the memory areas (I, Q, M, T, and C).

After the memory has been reset, the diagnostics buffer remains intact, as does the MPI address.

Using the Status Indicators

The status indicators (BUSF, INTF, EXTF, PS, BATTF, FRCE, RUN, and STOP) show basic information about the controller, such as the current operating mode or the presence of an error condition. You cannot change the status of the controller by clicking on the status indicators. For more information about the status indicators, see Section 5.1 and Table 5-2.

4.5 Understanding the WinLC Scan Cycle

As shown in Figure 4-10, the scan cycle of WinLC consists of four basic elements:

- WinLC writes the status of the process-image output table (the Q memory area) to the outputs.
- WinLC reads the states of the inputs into the process-image input table (the I memory area).
- WinLC executes the user program.
- WinLC waits until the minimum scan cycle time has elapsed and triggers the next scan cycle.

Note

For the first scan, WinLC does not write to the outputs. After the first scan, all other scans start by writing the process-image output table to the outputs.

The time that elapses between the end of one scan and the minimum scan cycle time (which starts the next scan) is the “sleep time.” During this “sleep time,” the Windows NT operating system executes any other tasks required.

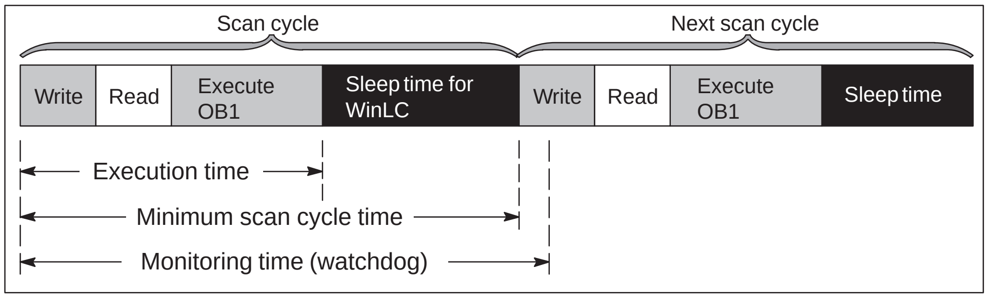


Figure 4-10 Elements of the WinLC Scan Cycle

Configuring the Elements of the Scan Cycle

You use the hardware configuration tool (“Cycle/Clock Memory” tab: see Section 5.5) of the STEP 7 programming software to enter the value for the minimum scan cycle time and the scan cycle monitoring time (watchdog). These values are stored as one of the default settings of the hardware configuration for WinLC.

Do not set the minimum scan cycle time to be longer than the scan cycle monitoring time (the watchdog time). The maximum value for the scan cycle monitoring time is 6 seconds; the minimum value for the scan monitoring time must be more than the minimum scan time.

You can also use the CPU panel to modify (or “tune”) the minimum scan cycle time; however, changes made to the minimum scan time with the CPU panel are only temporary and are replaced by the default values (in the hardware configuration) when you change the operating mode of the WinLC from STOP to RUN.

For more information about using the CPU panel to modify the sleep time or minimum scan cycle time, see Section 4.6.

Note

WinLC executes the cyclic interrupt OB (OB35) and other OBs at a fixed interval, independent of the scan cycle and of the execution of the user program in OB1. You must allow sufficient time not only for the execution of OB1 and for the sleep time, but also for the execution of other OBs.

For more information about OB35, refer to Section 5.5 and also to the *System Software for S7-300 and S7-400 System and Standard Functions Reference Manual*.

The following situations can increase the execution time of OB1:

- WinLC executes other OBs (such as OB20 and OB35) with higher priorities than OB1.
- You use the STEP 7 programming software to monitor and debug the user program.
- You use a variable table (VAT) with the STEP 7 programming software to display the status of the user program.
- Another application that is running on your computer has a higher priority under the Windows NT operating system.
- Interaction with HMI interfaces such as WinCC (Windows Control Center) or with the ActiveX controls supplied with the Computing software can affect the execution time of WinLC.

4.6 Tuning the Operation of WinLC

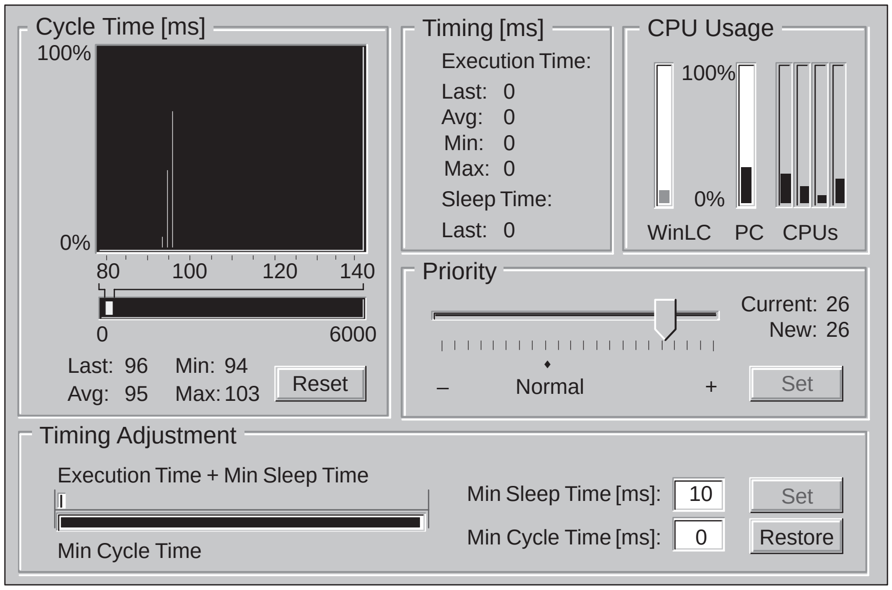
The CPU panel provides a panel for tuning the operation of WinLC. Using the tuning panel, you can adjust the minimum cycle time, minimum sleep time, and the Windows NT priority for WinLC.

Using the Tuning Controls to Modify WinLC Operations

The tuning panel (shown in Figure 4-11) allows you to modify elements of the WinLC scan and to set the priority level for Windows NT to execute the WinLC software:

- Priority: sets the level of priority for the execution of WinLC by the Windows NT operating system. Setting the priority higher for WinLC means that the operating system responds to WinLC before executing lower-priority tasks.
- Timing Adjustment: allows you to enter new values for either the sleep time or the minimum cycle time. After you enter the new value in the corresponding field, you can monitor the effect on the execution of WinLC. (You can restore the cycle time and sleep time values by clicking on the “Restore” button instead of clicking on the “Set” button.) To enter the new sleep time and cycle time values, click on the “Set” button. The panel then stores these values for the controller.

Using the tuning panel to enter a new value for either sleep time or scan cycle (cycle time) does not change the configured value stored in the SDBs. After you have determined the optimum values, use the STEP 7 programming software to change the hardware configuration. See Section 4.2 and Section 5.5.



Last: 96 Min: 94
Avg: 95 Max: 103

Timing [ms]
Execution Time:
Last: 0
Avg: 0
Min: 0
Max: 0
Sleep Time:
Last: 0

Current: 26
New: 26

Min Sleep Time [ms]: 10
Min Cycle Time [ms]: 0

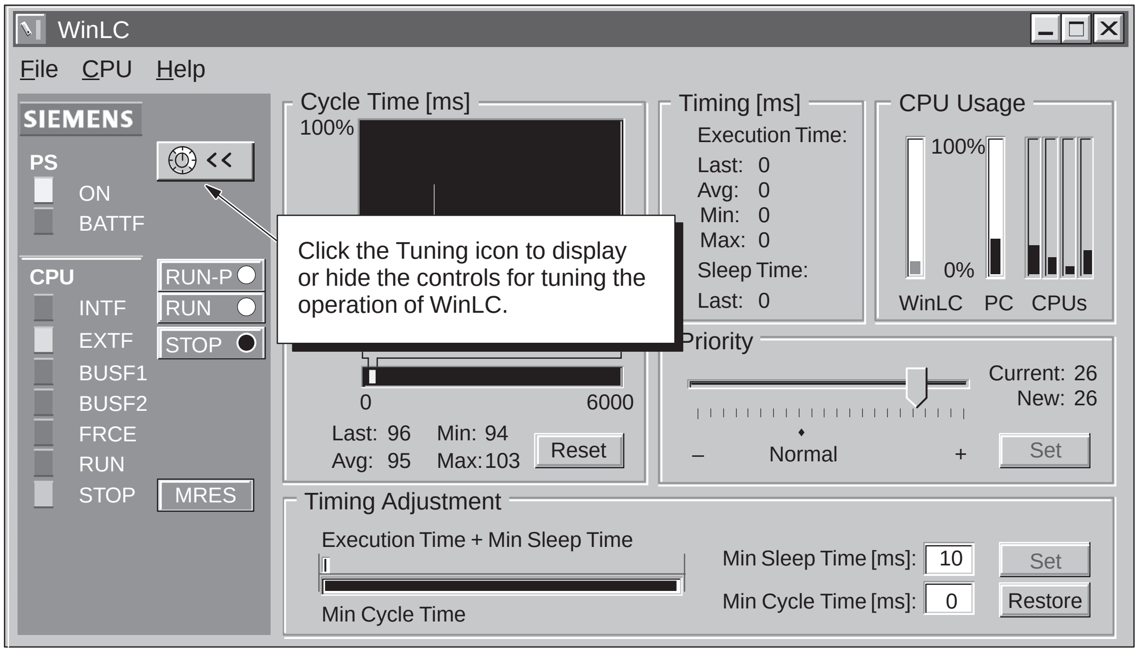
Figure 4-11 Tuning Panel for Adjusting the Operation of WinLC

The tuning panel also provides the following status information:

- Cycle Time (ms): provides a histogram of execution times (in ms) of the scans. This histogram tracks minimum (shortest) and maximum (longest) execution times, as well as the percentage of scans that fall in various ranges of execution times. Clicking on the "Reset" button deletes the historical data and starts a new histogram.
- Timing (ms): displays the execution time and the sleep time (in ms) for the last scan. It also displays the average execution time, the minimum (shortest) execution time, and the maximum (longest) execution time.
- CPU Usage: displays the percentage of the computer's processor resources that are allocated to WinLC. It also displays the total percentage for the processor capability of the computer (PC), and includes individual scales for each CPU of multi-processor systems.

Displaying the Tuning Interface

As shown in Figure 4-12, clicking on the Tuning icon of the CPU panel or selecting the **CPU ▸ Tuning Panel** menu command displays the tuning panel. If you have enabled password control, you must enter the password in the dialog box before the tuning panel appears. (For information about setting passwords, see Section 4.9.) After you have finished tuning WinLC, click on the tuning icon to hide the tuning panel.



Click the Tuning icon to display or hide the controls for tuning the operation of WinLC.

Last: 96 Min: 94
Avg: 95 Max: 103

Execution Time:
Last: 0
Avg: 0
Min: 0
Max: 0
Sleep Time:
Last: 0

Current: 26
New: 26

Min Sleep Time [ms]: 10
Min Cycle Time [ms]: 0

Figure 4-12 Displaying or Hiding the Tuning Panel of WinLC

4.7 Running the WinLC Controller

Closing the CPU panel does not shut down WinLC: you must manually shut down the WinLC controller or turn off the computer.

If you do not run WinLC as an NT service, the CPU panel allows you to start and stop WinLC. An Autostart feature allows you to start WinLC back up in the same operating mode (STOP, RUN, or RUN-P) that it was in before it was shut down.

Note

You must have administration privileges to register WinLC as a service. When you run WinLC as an NT service, you start or stop WinLC either from the “Services” dialog box or by turning your computer on or off. The CPU panel does not start or stop WinLC.

To access the “Services” dialog box, use the **Start ▸ Settings ▸ Control Panel** menu command to open the Windows NT control panel, then click on the “Services” icon.

Registering and Unregistering WinLC as an NT Service

The CPU panel provides a menu command for removing WinLC from the registry of NT services. See Figure 4-13. By unregistering WinLC, you can start or shut down the WinLC controller functions without having to turn the computer on or off. However, this also means that WinLC does not automatically start running whenever you turn on your computer.

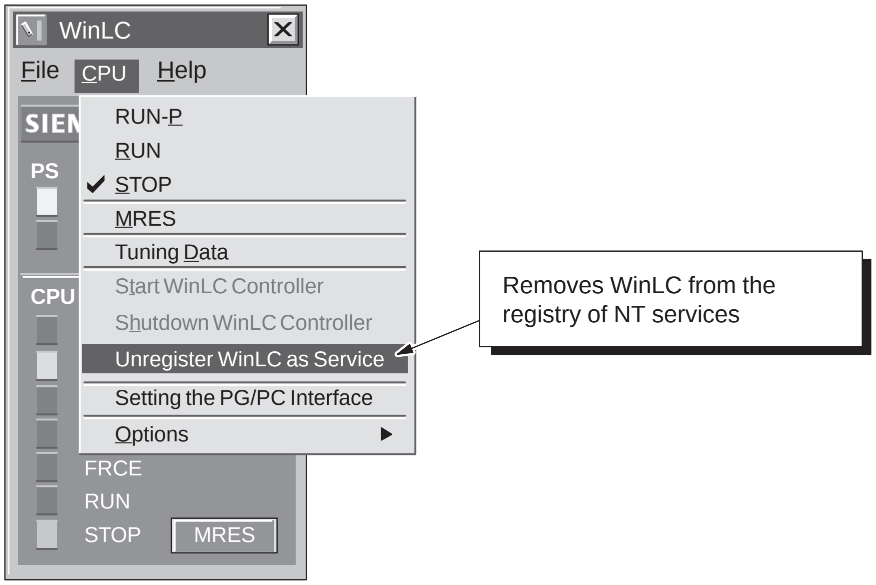

WinLC
File
CPU
Help
SIEN
PS
CPU
RUN-P
RUN
STOP
MRES
Tuning Data
Start WinLC Controller
Shutdown WinLC Controller
Unregister WinLC as Service
Setting the PG/PC Interface
Options
FRCE
RUN
STOP
MRES
Removes WinLC from the registry of NT services

Figure 4-13 Unregistering WinLC as an NT Service

Shutting Down and Starting the WinLC Controller

Closing the WinLC window does not shut down the WinLC controller: you must either change the CPU to STOP mode, manually shut down the WinLC controller, or turn off the computer.

Note

If WinLC is not running as an NT service, you can use the CPU panel to start or shut down the operation of the WinLC controller.

To shut down the WinLC controller, select the **CPU ▸ Shutdown WinLC Controller** menu command from the CPU panel (as shown in Figure 4-14). The WinLC controller then stops its operations.

To start the WinLC controller, select the **CPU ▸ Start WinLC Controller** menu command from the CPU panel.

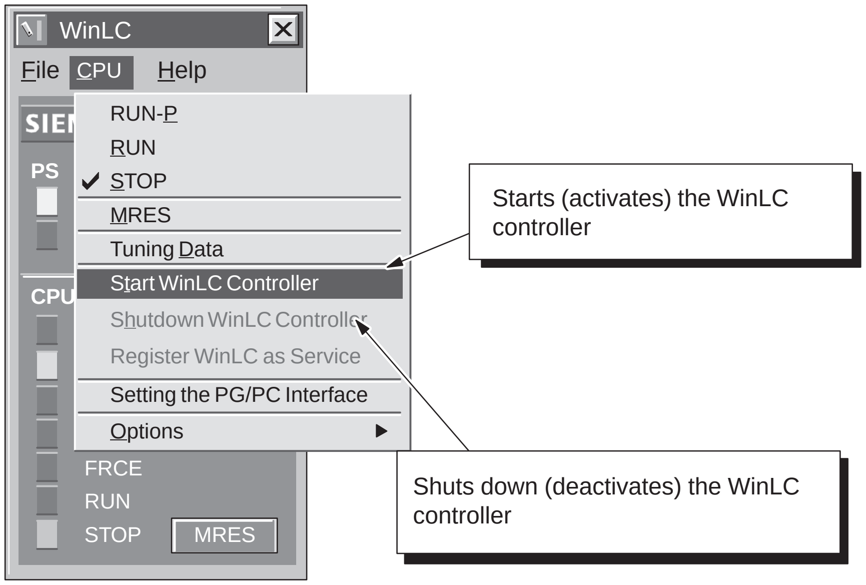

WinLC
File CPU Help
SIE
PS
CPU
RUN-P
RUN
STOP
MRES
Tuning Data
Start WinLC Controller
Shutdown WinLC Controller
Register WinLC as Service
Setting the PG/PC Interface
Options
FRCE
RUN
STOP
MRES
Starts (activates) the WinLC controller
Shuts down (deactivates) the WinLC controller

Figure 4-14 Shutting Down the WinLC Controller

Selecting the Autostart Feature

WinLC includes an Autostart feature that defines how WinLC responds to shutting down and restarting. Based on the parameters shown in Table 4-1, WinLC starts in the specified operational mode.

You use the "Customize" dialog box to enable or disable the Autostart feature.

Table 4-1 Autostarting the WinLC controller

If the WinLC controller was running at shutdown ...	And the Autostart feature is selected ...	Then WinLC starts with this operating mode
No	No	STOP mode
No	Yes	STOP mode
Yes	No	STOP mode
Yes	Yes	RUN mode

Use the following procedure to enable the Autostart feature of WinLC:

1. As shown in Figure 4-15, select the **CPU** ▸ **Options** ▸ **Customize** menu command to display the "Customize" dialog box.

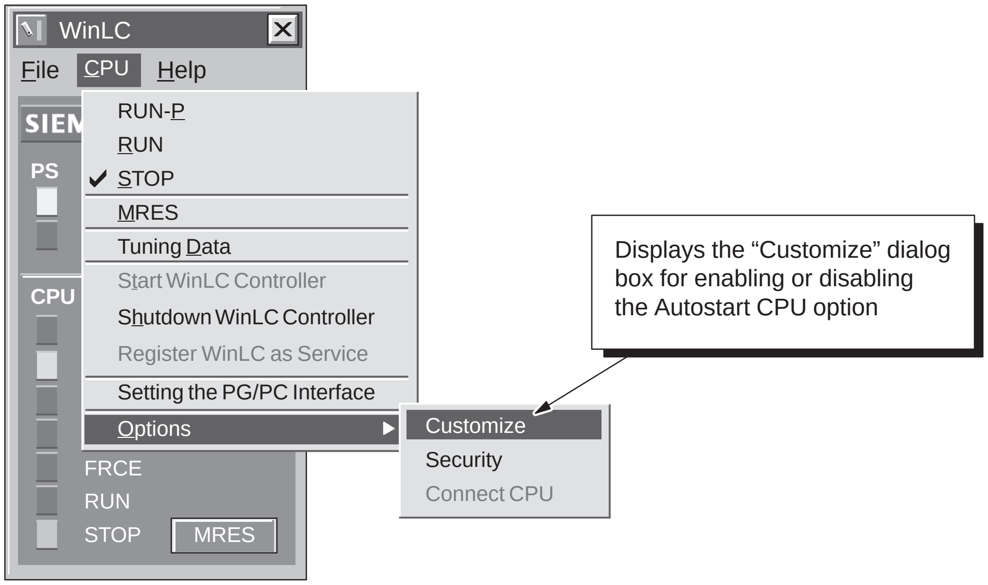
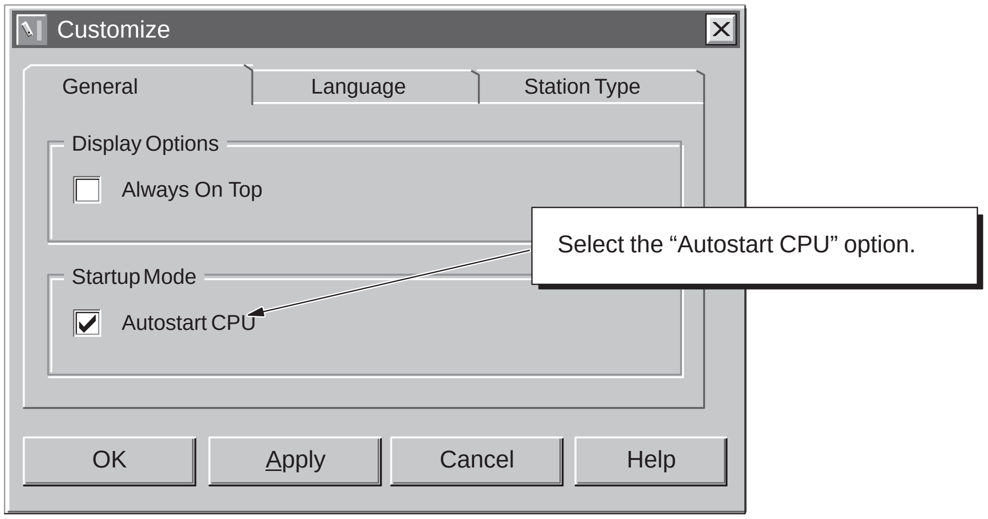

WinLC
File
CPU
Help
SIEM
PS
CPU
RUN-P
RUN
✓ STOP
MRES
Tuning Data
Start WinLC Controller
Shutdown WinLC Controller
Register WinLC as Service
Setting the PG/PC Interface
Options
Customize
Security
Connect CPU
FRCE
RUN
STOP
MRES
Displays the "Customize" dialog box for enabling or disabling the Autostart CPU option

Figure 4-15 Accessing the "Customize" Dialog Box

2. In the “Customize” dialog box, select the “General” tab and select the “Autostart CPU” option. See Figure 4-16.
3. Click on the “Apply” button to enable the Autostart feature.
4. Click on the “OK” button to close the “Customize” dialog box.



Display Options

☐ Always On Top

Startup Mode

☑ Autostart CPU

Select the “Autostart CPU” option.

Figure 4-16 Selecting the Autostart Feature

4.8 Selecting the Language for WinAC

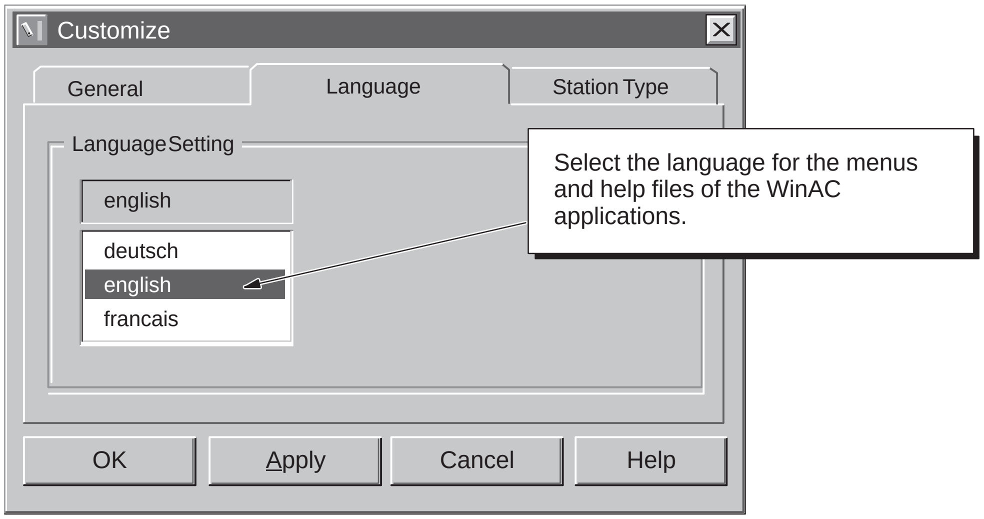
WinAC provides three languages for the software and help: German, English, and French. The menus and help for the WinLC are displayed in the language selected. You can change the language from the CPU panel of WinLC.

Use the following procedure to change the language for WinAC:

1. Select the **CPU ▸ Options ▸ Customize** menu command to display the “Customize” dialog box.
2. In the “Customize” dialog box, select the “Language” tab.
3. Select the language for the CPU panel (German, English, or French). See Figure 4-17.
4. Click on the “Apply” button to change the language.
5. Click on the “OK” button to close the “Customize” dialog box.

Note

The change in language for WinLC does not become effective until you restart the WinAC applications.



Select the language for the menus and help files of the WinAC applications.

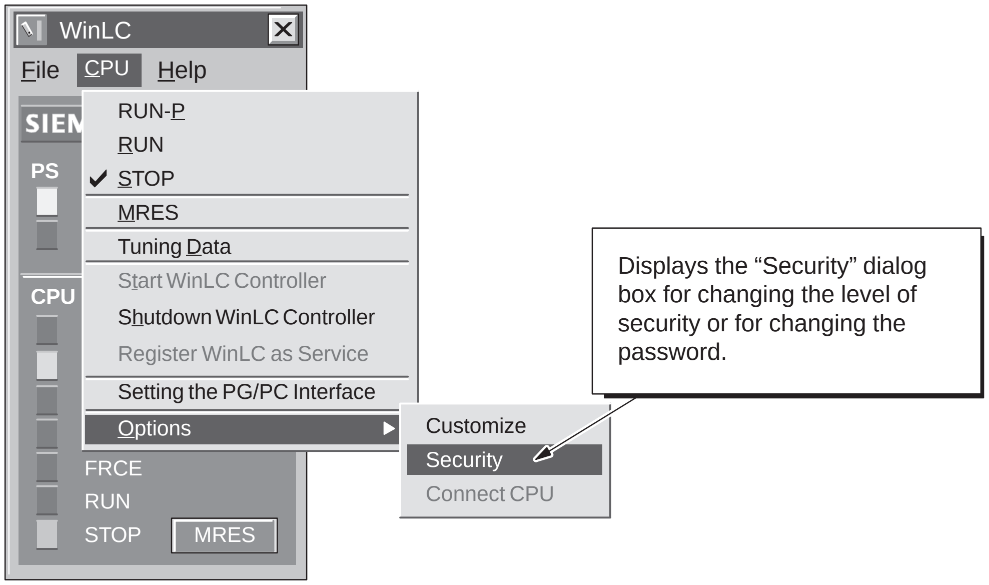
Figure 4-17 Selecting the Language for the CPU Panel and Help Files

4.9 Creating Levels of Security for Access to WinLC

You can set the CPU panel to create levels of security and limit access to WinLC:

- Select the security level: You can set WinLC to request confirmation or enable password-protection before allowing any changes.
- Configure the password to be valid for a specific amount of time: You can set a specific length of "password free" time in which the user is not required to enter another password when making changes. This length of time can be up to 23 hours and 59 minutes after the user has initially entered the password.
- Change the password: You can easily change the password with the "Change Password" dialog box.

To access the "Security" dialog box, select the **CPU ▸ Options ▸ Security** menu command from the CPU panel. See Figure 4-18.



Displays the "Security" dialog box for changing the level of security or for changing the password.

Figure 4-18 Accessing the "Security" Dialog Box

Warning

Running the WinLC controller without confirmation or password protection increases the risk that the operating mode could be changed inadvertently. This could cause erratic behavior of the process or machinery being controlled, which could cause damage to equipment or injury to personnel.

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

Changing the Security Level for WinLC

You can create levels of security and limit access to the controller. WinLC provides the following levels:

- None: No confirmation or password is required to access WinLC.
- Confirmation: Any change made with the CPU panel (such as changing operating mode or tuning the operations) requires confirmation by acknowledging a message box.
- Password: Any change made with the CPU panel (such as changing operating mode or tuning the operations) requires that the user enter a password.

Use the following procedure to change the security level for WinLC:

1. Select the **CPU ▸ Options ▸ Security** menu command.
2. In the “Access verification” dialog box, enter the password for WinLC and click on the “OK” button. See Figure 4-19. (If the security level is set to “None” or no password has been configured, simply click on the “OK” button.)

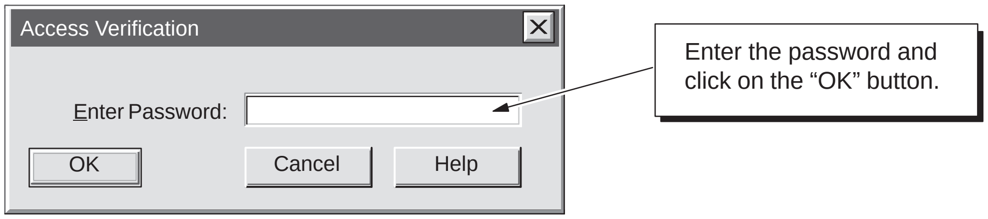


Figure 4-19 Entering the Password for WinLC

3. In the “Security” dialog box (Figure 4-20), click on the option for password (security level).
4. Click on the “OK” button to enter the changes and close the “Security” dialog box.

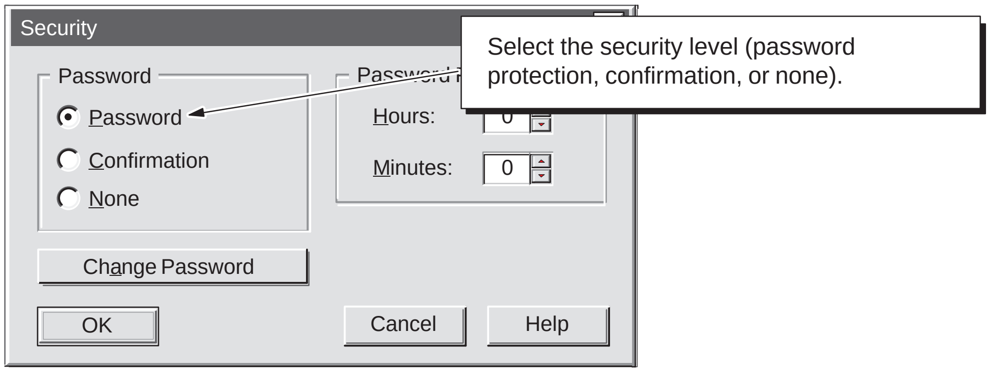
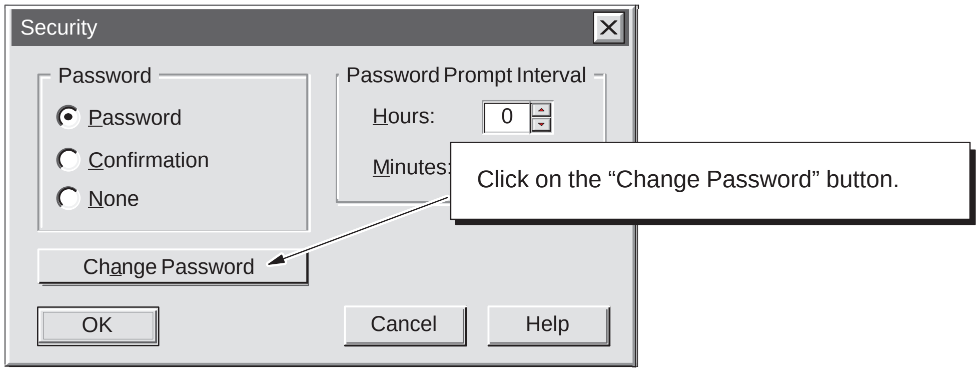


Figure 4-20 Setting the Security Level for WinLC

Creating or Changing the Password for WinLC

The “Security” dialog box allows you to create or to change the password for WinLC. Use the following procedure for creating or changing a password:

1. Select the **CPU** ▸ **Options** ▸ **Security** menu command.
2. In the “Access verification” dialog box, enter the password for WinLC and click on the “OK” button. (If the security level is set to “None” or no password has been configured, simply click on the “OK” button.)
3. In the “Security” dialog box, click on the “Change Password” button. See Figure 4-21.



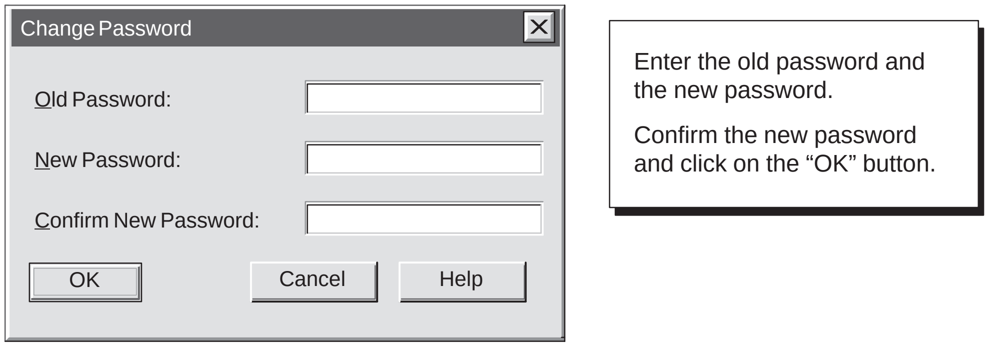
Password [✓ Password] Confirmation None

Hours: 0

Click on the “Change Password” button.

Figure 4-21 Accessing the “Change Password” Dialog Box

4. As shown in Figure 4-22, enter the following information in the “Change Password” dialog box:
   – In the “Old Password” field, enter the text string for the previous password.
   – In the “New Password” field, enter the new text string for the password.
   – In the “Confirm New Password” field, enter the text string for the new password.



Enter the old password and the new password.

Confirm the new password and click on the “OK” button.

Figure 4-22 Changing a Password for WinLC

5. Click on the “OK” button to change the password and return to the “Security” dialog box.
6. Make certain that the security level of WinLC is set to the “Password” option and click on the “OK” button to accept the changes and close the “Security” dialog box.

Note

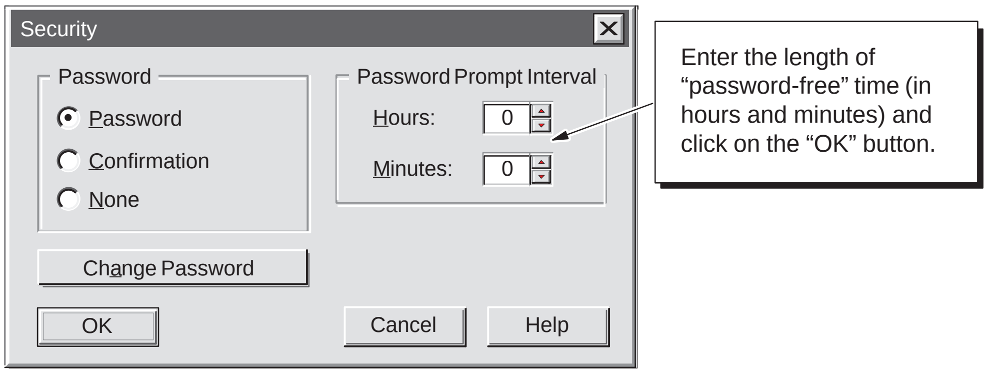
If you create a password, but set the security level to “None” (disabling the password), you will still need to enter the configured password before you can access the “Security” dialog box again.

Making the Password Valid for a Specific Length of Time

By configuring the “validity” of the password, you set a specific length of “password-free” time during which the user is not required to enter another password when making changes. This length of time can be up to 23 hours and 59 minutes after the user has initially entered the password.

Use the following procedure to configure the validity for the password:

1. Select the **CPU ▸ Options ▸ Security** menu command.
2. In the “Access verification” dialog box, enter the password for WinLC and click on the “OK” button. (If the security level is set to “None” or no password has been configured, simply click on the “OK” button.)
3. In the “Security” dialog box, enter the length of time for the password to be valid. See Figure 4-23.
   – Enter up to 23 hours in the “Hours” field.
   – Enter up to 59 minutes in the “Minutes” field.
4. Click on the “OK” button to enter the length of validity for the password.
5. Make certain that the security level of WinLC is set to the “Password” option, and click on the “OK” button to accept the changes and close the “Security” dialog box.


Security

Password: [✓ Password] Confirmation None

Password Prompt Interval
Hours: 0
Minutes: 0

Enter the length of “password-free” time (in hours and minutes) and click on the “OK” button.

Figure 4-23   Configuring the Password Validity

4.10 Saving and Restoring Your User Program

You can save the Load memory (user program) to an archive file. You can use this archive file like a memory cartridge: you can easily restore the user program from the archive file.

If you load your user program by restoring an archive file, the archive file is not automatically restored again after a memory reset (MRES) operation (unlike the behavior of an EPROM in a hardware PLC). You can use the **File ▸ Restore** menu command to perform this operation manually.

Creating an Archive File

As shown in Figure 4-24, you create an archive file by selecting the **File ▸ Archive** menu command from the CPU panel. A dialog box allows you to save the archive file under a specific name. This allows you to store different archive files.

The archive file contains the user program and the hardware configuration (SDBs).

Restoring the Archive File

When you restore the archived file, you reload the user program and the hardware configuration (SDBs). To reload a user program from an archive file, follow these steps:

1. Click on the “STOP” button to place the controller in STOP mode.
2. Press the “MRES” button to perform a memory reset.
3. Select the **File ▸ Restore** menu command from the CPU panel (as shown in Figure 4-24).
4. Select the specific archive file to reload.

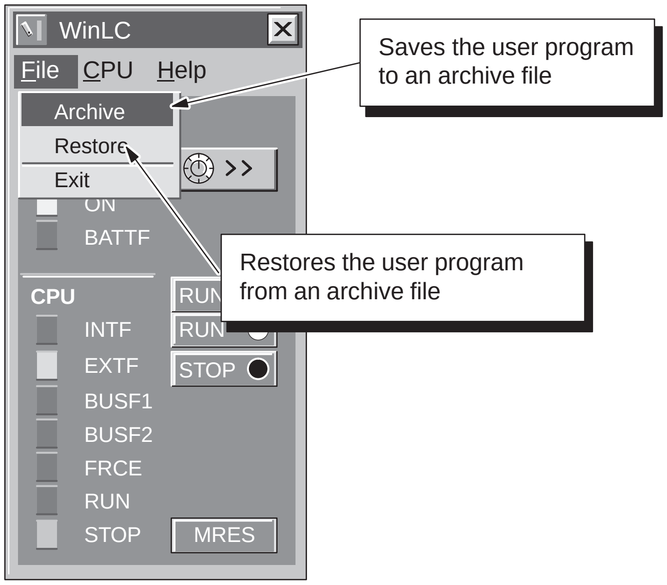


Figure 4-24   WinLC Archive and Restore Commands

Operations of WinLC 5

Chapter Overview

WinLC is a programmable logic controller (PLC) that runs on your computer. It communicates with the distributed (remote) I/O over a PROFIBUS-DP network. For more information about using PROFIBUS-DP, see Chapter 6 and the *SIMATIC NET PROFIBUS User Manual*.

This chapter describes the basic operation of WinLC and includes the following information:

- Elements of the WinLC interface. For additional information, see Chapter 4 and the online help of the WinLC software.
- Memory reset function (MRES) of the WinLC memory
- Real-time clock. For more information, see the *System Software for S7-300 and S7-400 System and Standard Functions Reference Manual*.
- Configuration of the WinLC parameter blocks. For more information, see the *STEP 7 User Manual* and the online help of the STEP 7 software.

5.1 Mode Selector and Status Indicators of the CPU Panel

The CPU panel corresponds to the faceplate of the S7 CPU modules. As shown in Figure 5-1, the CPU panel contains buttons for changing the operating mode of WinLC, a button for resetting the memory areas, and status indicators. For detailed information about resetting the WinLC memory, see Section 5.2.

Note

Indicators that are not applicable for WinLC are “grayed-out.”

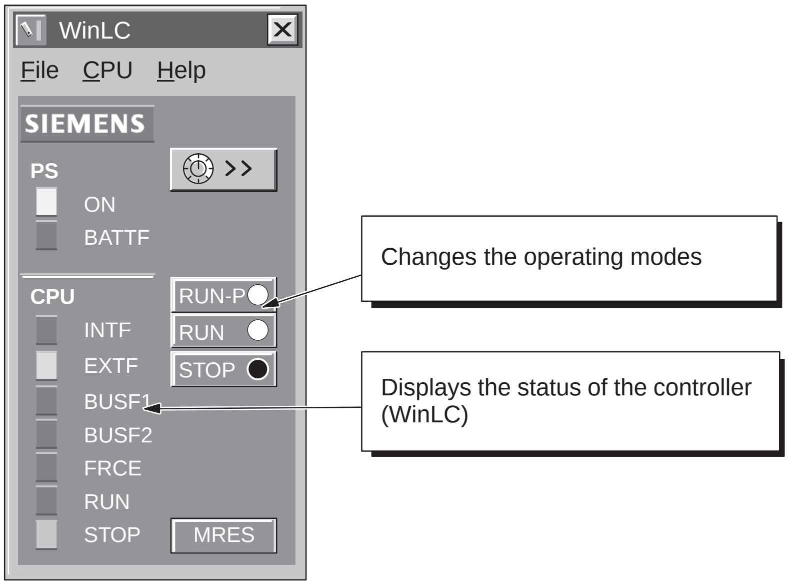


Figure 5-1 Mode Selector Buttons and Status Indicators of the CPU Panel

Selecting the Operating Mode

The RUN, RUN-P, and STOP buttons on the CPU panel correspond to the different operating modes of the controller. Table 5-1 describes the operating modes. Clicking on one of these buttons places WinLC into the selected operating mode.

To allow an external source, such as the STEP 7 programming software, to change the operating mode of WinLC, select RUN-P mode. If the operating mode is changed by the external software, the selected button on the CPU panel does not change, but the status indicators reflect the actual operating mode of WinLC.

Table 5-1 Operating Modes of WinLC

Mode	Description
RUN-P	WinLC executes the user program. When WinLC is in RUN-P mode (RUN-PROGRAM mode), you can: • Upload a program from WinLC to your computer or programming device • Download a program to WinLC • Download individual blocks to WinLC • Use external software (such as STEP 7) to change the operating mode of WinLC
RUN	WinLC executes the user program. When WinLC is in RUN mode, you can upload a program from WinLC to your computer or programming device; however, you cannot download a program to WinLC.
STOP	WinLC does not execute the user program. When the controller is in STOP mode, you can: • Upload a program from WinLC to your computer or programming device • Download a program to WinLC

Running WinLC without Valid Authorization

If you lose the authorization for the software, WinLC will continue to run, although with less functionality. You will still be permitted to change from RUN mode to STOP mode or from STOP mode to RUN mode, but you cannot change WinLC to RUN-P mode. WinLC continues to execute the user program, but you cannot modify the user program, and you cannot download a new program or new blocks to WinLC.

Using the Status Indicators

The status indicators on the CPU panel show basic information about WinLC, such as the current operating mode or the presence of an error condition. Table 5-2 describes the different status indicators for the CPU panel. You cannot change the status of WinLC by clicking on the status indicators.

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

During a restart, both the RUN and STOP indicators are turned on: the RUN indicator flashes, and the STOP indicator is on during the restart; when the STOP indicator turns off, the outputs are enabled.

Table 5-2 Status Indicators

Indicator	Description
ON	Power supply. Always on for WinLC.
BATTF	Battery fault. Always off for WinLC.
INTF	This indicator lights up (solid) to show error conditions within the controller, such as programming errors, firmware errors, arithmetic errors and timer errors.
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	These indicators light up (either solid or flashing) to identify fault conditions in the communication with the distributed I/O. See Table 6-5. Since WinLC supports only 1 PROFIBUS-DP network, BUSF1 is the only active indicator; BUSF2 is not applicable for WinLC.
FRCE	This indicator lights up (solid) to show that a force request is active. Not applicable for WinLC.
RUN STOP	Lights up (solid) to show the operating mode (RUN or STOP) When RUN is flashing and STOP is lighted (solid): • The controller is executing a restart. (Run light blinks with 2 Hz.) • The user program has reached a break point. (Run light blinks with 0.5 Hz.)
All status indicators are flashing	When all of the status indicators are flashing, WinLC has encountered an error condition that cannot be fixed by resetting the memory (MRES). To recover from this condition, you must perform the following tasks: 1. Shut down the WinLC controller. 2. Restart the WinLC controller. 3. Reset the memory (MRES). If WinLC is running as a service, you must use the Windows NT control panel to shut down and restart the WinLC controller.

5.2 Resetting the WinLC Memory

The CPU panel provides a MRES button for resetting the memory areas to the default values and deleting the user program from the Load memory and work memory areas.

You can also use STEP 7 to reset the WinLC memory; however, WinLC must already be in STOP mode.

You normally reset the memory areas before downloading a new program to WinLC or restoring an archive file. You also reset the memory if the STOP indicator on the CPU panel is flashing to alert you to the following conditions:

- Errors were detected in the work memory area.
- The size of the user program exceeded the work memory area.

Using the MRES Button to Reset the Memory Areas

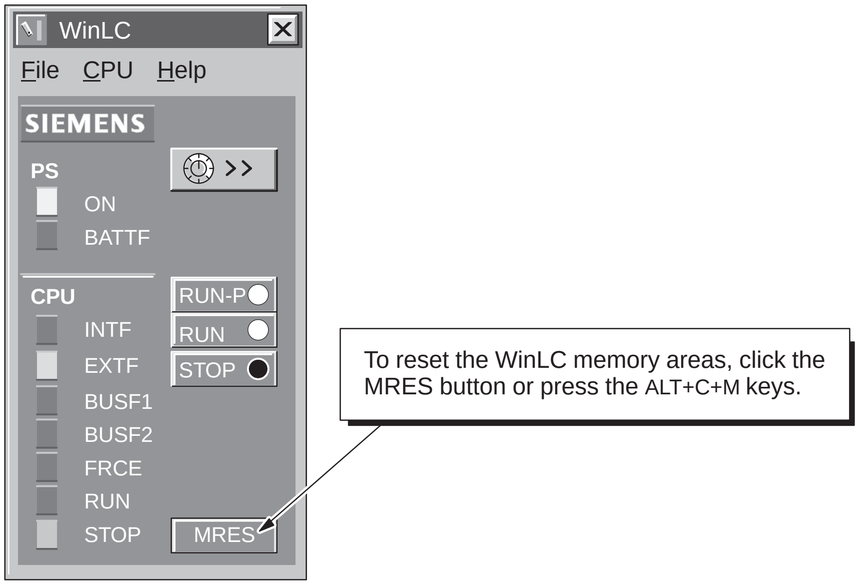
The MRES button performs a memory reset on the memory areas. See Figure 5-2. Clicking the MRES button places WinLC into STOP mode and performs the following tasks:

- WinLC deletes the entire user program from both the work memory area and the load memory area. This includes the data blocks (DBs).
- WinLC deletes the backup memory and resets the memory areas (I, Q, M, T, and C) to 0.

After the memory has been reset, the diagnostics buffer remains intact, as does the MPI address.

Note

To reset the memory without using the mouse, press the ALT+C+M keys.



To reset the WinLC memory areas, click the MRES button or press the ALT+C+M keys.

Figure 5-2 Resetting the WinLC Memory with the CPU Panel

5.3 Using the Diagnostic Information Stored in WinLC

As described in Section 5.1, the CPU panel provides indicators that display information about the status of WinLC. In addition to the status information, you can use the STEP 7 programming software to read diagnostic and operational information.

STEP 7 also provides additional tools for testing and monitoring a program running on WinLC.

Monitoring the Diagnostic Information

When WinLC encounters an error condition while in RUN mode (executing the user program), WinLC turns on the SF (system fault) indicator and writes one or more entries to the diagnostics buffer. Based on the type of error and the organization blocks (OBs) that were downloaded with the program, WinLC then either goes to STOP mode or executes the appropriate OB, which allows your program to react to the error condition. For more information about the OBs which are available for WinLC, see Section B.2.

WinLC stores diagnostic information in different registers and stacks. (You can access this information from the **Accessible Nodes** menu command.) Table 5-3 lists the types of information which can be viewed by using the tools provided by STEP 7. For more information about viewing and using the diagnostic information, see the online help for STEP 7 or the *STEP 7 User Manual*.

Table 5-3 Diagnostic Information Provided by WinLC

Information	Description
Communication	Displays information about the transmission rates, communication connections, communication load, and the maximum frame size for messages on the communication bus.
Cycle time	Displays the durations for the longest, shortest, and last scan cycle.
Diagnostic buffer	Displays the contents of the diagnostics buffer, including a description of the event and the time and date that the event occurred.
General	Displays general information about WinLC, such as the project path, the version number, and the order number
Memory	Displays the current utilization of the Work memory and the Load memory of WinLC.
Performance data	Displays the memory configuration and the valid addresses for the controller. Clicking on the “Blocks” button displays all of the blocks (OBs, SFBs, SFCs, FBs, FCs, and DBs) which are available (including all priority classes).
Scan cycle time	Displays information about the cycle time of the user program, including the longest cycle time, the shortest cycle time, the minimum cycle time, and the last cycle time.
Stacks	Displays the contents of the B Stack (block stack), the I Stack (interrupt stack), and the L Stack (local data stack)
Time system	Displays information about the current time, the operating hours, and the synchronization of the system clock.

Monitoring and Modifying the Variables in the User Program

STEP 7 provides tools for monitoring the status of a user program running on WinLC. You can also use STEP 7 to modify the value of the process variables used by your program. For more information about monitoring and modifying the process variables in a program, see the online help for the STEP 7 programming software or the *STEP 7 User Manual*.

WinLC allows you to use the STEP 7 tools to perform the following tasks:

- Monitoring variables: You can monitor the status of the different process variables used in your program. You can view the status of your program either with a status chart or by turning on the status option in the Program Editor.
- Modifying variables: You can modify a process variable by entering a specific value. By changing the values of the process variables, you can monitor the behavior of your program. You can modify variables with a status chart.

Viewing a Status Block

You can monitor a block with regard to the program sequence for support in startup and troubleshooting. The status block allows you to monitor the contents of registers, such as the address register, the status register, and the DB registers, while WinLC is executing the user program.

5.4 System Clock Supported by WinLC

WinLC supports a real-time clock. The user program being executed by the controller can access this information by using different SFCs. The real-time clock is based on the hardware clock of the computer that is running WinLC.

You can use the STEP 7 programming software to set the system clock of the controller to a time that is different from that of the hardware clock of your computer. If you close the WinLC application, this difference from the hardware clock is maintained: when you open the WinLC application again, the system clock of the controller reflects the passage of time while the WinLC application was closed.

Using the Real-Time clock

The default setting for the system clock is the default date and time for the hardware clock of your computer.

You can also use SFC0 (SET_CLK) and SFC1 (READ_CLK) to set and read the system clock. For more information about using these SFCs, see the *System Software for S7-300 and S7-400 System and Standard Functions Reference Manual*.

5.5 Configuring the Operational Parameters of WinLC

STEP 7 provides the tool for configuring the characteristics and behavior of WinLC. You use the Hardware Configuration tool to display a dialog that configures the operational characteristics for WinLC. This configuration is then stored in SDB0. Table 5-4 lists the different parameters that can be configured. For more information about configuring the operational parameters, see the *STEP 7 User Manual*.

After you download SDB0, the controller uses the configured parameters for the following events:

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

Table 5-4 WinLC Configuration Parameters Provided by STEP 7

Parameters	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 scan cycle time and the size of the process image) 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
Interrupts	Configures the operation of the time-of-day interrupts (OB10)
Time-Of-Day Interrupts	Defines the priority for the hardware interrupts (OB40), the time-delay interrupts (OB20), and the asynchronous error interrupts (OB82, OB83, OB85, and OB86)
Retentive Memory	Defines the memory areas (M, T, and C) as well as the DBs 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 (OB35, 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

Configuring the Startup Characteristics

Using the “Startup” tab of the STEP 7 Hardware Configuration, you can configure WinLC to perform certain tasks before going to RUN mode. Table 5-5 lists the parameters for configuring the startup characteristics.

Table 5-5 Parameters for the Startup Characteristics

Parameter	Description	Range	Default
Startup on setpoint configuration not equal to actual configuration	Reserved for future use.	Not applicable	Yes
Startup after Power On	WinLC provides a cold restart (OB102) and a warm restart (OB100).	Warm restart Cold restart	Warm restart

Configuring the Clock Memory

Using the “Cycle/Clock Memory” tab of the STEP 7 Hardware Configuration, you can configure a byte of the bit memory (M) area to function as a “clock memory.” Table 5-6 lists the parameters and ranges for configuring a clock memory.

Table 5-6 Parameters for Configuring a Byte as 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 maximum for M memory	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.) Figure 5-3 shows the frequencies of the different bits for the byte used as clock memory.

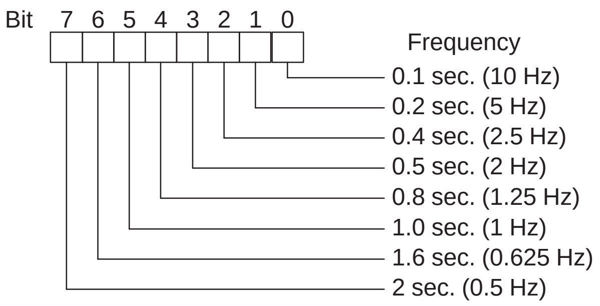


Figure 5-3 Clock Frequencies for the Memory Byte Configured as Clock Memory

Configuring the Scan Cycle

Using the “Cycle/Clock Memory” tab of the STEP 7 Hardware Configuration, you can configure WinLC to control certain aspects of the scan cycle. Table 5-7 lists the parameters for configuring the scan cycle. For more information about the scan cycle, see Section 4.6.

Note

The minimum cycle time of WinLC encompasses both the time required for executing the user 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 (program execution time 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.

Table 5-7 Parameters for Controlling the Scan Cycle

Parameter	Description	Range	Default
Scan cycle monitoring time	Enters the maximum time for the scan cycle plus the sleep time for the controller. This value must be larger than the value for the minimum scan cycle time. The following list gives a few examples of events that could cause the controller to exceed the limit on maximum cycle time: • Starting other PC applications • An increasing number of interrupts in the program • Processing an error in the user program	1 to 6000 ms	6000
Minimum scan time	Enters the minimum time for the scan cycle. This value includes both the execution time of the user program and the sleep time of WinLC. For more information about the scan cycle, see Section 4.6. 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).	0 to 6000 ms	0

Configuring the Retentive Areas of Memory

Using the “Retentive Memory” tab of the STEP 7 Hardware Configuration, you can 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:

- Memory bytes: up to 256 bytes (from MB0 to MB255)
- S7 timers: up to 128 timers (from T 0 to T 127)
- S7 counters: up to 64 counters (from C 0 to C 63)

Table 5-8 lists the parameters for configuring the retentive memory areas.

In the event of a transition from STOP mode to RUN mode, WinLC does not reset the values which are stored in the timers, counters, and memory bytes that are configured to be retentive. All DBs are retentive.

In the event of a power failure while the controller is running, the current values are lost. If you close the WinLC software before the power failure, the values are retained according to the configured parameters shown in Table 5-8.

Note

DBs that were created by SFC22 (CREATE_DB) are not retained following a cold restart.

Table 5-8 Parameters for Configuring the Retentive Memory

Parameter	Description	Range	Default
Memory Bytes	Enters the number of memory bytes to be retained (starting from MB0)	0 to 256	16
S7 Timers	Enters the number of S7 timers to be retained (starting from T 0)	0 to 128	0
S7 Counters	Enters the number of S7 counters to be retained (starting from C 0)	0 to 64	8

Configuring the Time-of-Day Interrupt

WinLC supports one time-of-day interrupt (OB10). Using the “Time-of-Day” tab of the STEP 7 Hardware Configuration, you can configure OB10. Table 5-9 lists the parameters for the time-of-day interrupt.

Table 5-9 Parameters for Configuring 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 date and time	01.01.94 00:00:00

Configuring the Interrupts

Using the “Interrupts” tab of the STEP 7 Hardware Configuration, you can configure the priority class for some of the interrupt OBs supported by WinLC. Table 5-10 lists the parameters for the different interrupts.

WinLC has the following restriction: you cannot change the priority class for OB20 (time-delay interrupt).

Table 5-10 Parameters for Configuring the Priority Class of the Interrupts

Interrupt	Description	Range	Default
Time-Delay OB20	You cannot change the priority class for the time-delay interrupt	0, 2 to 24	3
Asynchronous Error OB80 to OB87	Defines the priority class for the asynchronous error interrupts	OB80: 26 OB81 to OB87: 24 to 26	OB80: 26 OB81 to OB87: 26

Configuring the Cyclic Interrupt

WinLC supports one cyclic interrupt (OB35). Using the “Cyclic Interrupts” tab of the STEP 7 Hardware Configuration, you can configure time interval for executing OB35. Table 5-11 lists the parameters for the cyclic interrupt.

Table 5-11 Parameters for Configuring the Cyclic Interrupt

Parameter	Description	Range	Default
Priority	Determines the priority class of the cyclic interrupts (OB35, OB36)	0, 2 to 24	OB35: 12 OB36: 13
Execution	Determines the time interval (in milliseconds) for executing the cyclic interrupts	0 to 60000	OB35: 100 OB36: 50
Phase Offset	Defines an amount of time that the start of a cyclic interrupt can be delayed in order to allow another cyclic interrupt to finish	0 to 60000	0

Based on the interval that was configured, WinLC starts the execution of OB35 at the appropriate interval. For best results, choose an interval greater than 10 ms. Selecting an interval of less than 10 ms can cause OB35 not to be executed at the scheduled time. The causes for OB35 not being executed can include:

- The program in OB35 takes longer to execute than the interval allows.
- Programs in other priority classes frequently interrupt or take longer to execute, which causes the controller not to execute OB35 at the scheduled time.
- A programming device performs some task or function that causes the controller not to execute OB35 at the scheduled time.

The sleep time of the WinLC scan cycle (see Section 4.6 and Figure 4-10) does not affect the execution of OB35: WinLC executes OB35 at the appropriate interval regardless of the amount of sleep time that you configure for the scan. (See Section 4.6.) Having OB35 run too frequently or require too much of the time allotted for the total scan could cause the watchdog timer to time out (calling OB80 or going to STOP mode).

Assigning the Parameters for Diagnostics

Using the “Diagnostics/Clock” tab of the STEP 7 Hardware Configuration, you can configure how WinLC responds to the various events that are recognized and evaluated during the execution of the user program. Table 5-12 lists the parameters for configuring the handling of diagnostic events.

WinLC can recognize certain diagnostic events, such as an error in the user program, a failed module, or an open circuit on the connector of a module. Events that are not transmitted to WinLC (such as a defective motor) must be handled by the user program by some error detection program for the process.

Table 5-12 Parameters for Configuring the Diagnostic Activities

Parameter	Description	Range	Default
Display Cause of STOP	Determines whether to send the last message (the most recent) in the diagnostic buffer to the registered display unit	Yes/No	Yes
Synchronization on MPI	Determines whether to use the WinLC clock to synchronize the clocks of other controllers: • None: no synchronization • As Slave: WinLC clock is synchronized from another clock	None or As Slave	None
Correction Factor (ms)	Compensates for a loss or gain in the clock time within a 24-hour period. Value is entered in milliseconds (1 second = 1000).	–99999 to 99999	0

Configuring the PROFIBUS-DP Network 6

Chapter Overview

The WinLC controller uses a PROFIBUS-DP network to communicate with the distributed I/O. The controller is the DP master station, and the I/O modules (for example, ET 200B or ET 200L) are the DP slave stations. An S7 CPU (such as the CPU 315–2 DP) can also function as an intelligent slave device.

You use the hardware configuration tools of the STEP 7 programming software to assign the addresses and other parameters for WinLC (DP–master) and the I/O (DP–slaves). For more information, see the online help for STEP 7 programming software and the *STEP 7 User Manual*.

For more information about DP communications and setting up PROFIBUS networks, refer to the *SIMATIC NET PROFIBUS User Manual*.

Section	Description	Page
6.1	Guidelines for Configuring the PROFIBUS-DP Network	6-2
6.2	Determining the Physical Layout of the Network	6-6
6.3	Assigning the Addresses for the Distributed I/O	6-8
6.4	Starting the PROFIBUS-DP Network	6-14

6.1 Guidelines for Configuring the PROFIBUS-DP Network

PROFIBUS-DP (Process Field Bus – Distributed Peripherals) is an industry standard for process communications with distributed peripherals. WinLC uses PROFIBUS–DP to connect to its distributed I/O. Digital, analog, and intelligent I/O (including field devices such as drives and valve terminals) can be installed as distributed I/O on the PROFIBUS-DP network.

The PROFIBUS-DP network connects the controller with the distributed I/O modules. The controller and the I/O modules are called “nodes” or “stations”: WinLC must be the master node (DP master) and the distributed I/O are the slave nodes (DP slaves). You can connect up to 125 DP slaves for WinLC. A network consists of one or more segments. Each segment can have up to 32 nodes (including repeaters on either end of the segment).

Device Types

Devices connected to a PROFIBUS–DP network are referred to as “nodes” or “stations”: A node may be a DP master (controlling) or a DP slave (controlled) node. For DP networks used by WinLC, WinLC is the master node and the distributed I/O devices are slave nodes.

Each node on a DP network must have a unique node address. Node addresses can be assigned in the range 0..125. You can connect up to 126 nodes on a DP network. Since WinLC counts as one of these nodes, this means that WinLC can control up to 125 DP slaves.

Note

WinLC supports a total of 125 DP slaves; however, the CP card used with WinLC may support fewer slaves. For example, the CP5412 card allows a maximum of 64 slaves.

A DP slave can consist of one or more modules. The modules may be integrated into the node (ET200B) or they may be separately installable (ET200M).

Cabling

From a cabling view point, a DP network consists of one or more segments, where segment is the bus line between two terminating resistors. The nodes are connected in series to a network segment. The first and last nodes of a segment must have a powered termination circuit switched to the ’on’ position whenever the network is operational. All other nodes of the segment must have their termination circuits switched to the ’off’ position.

Network segments are connected using repeaters. A DP network can have many segments as long as the following guidelines are observed:

- A maximum of ten segments can be connected in series. In other words, the signal path from any node on the network to any other node on the network must not pass through more than nine repeaters.
- No segment can have more than 32 nodes. The repeaters connected to a segment count in the node count for the segment.
- No segment can exceed the maximum cable length allowed for the baud rate used by the network.

Figure 6-1 shows a sample network consisting of a single segment with three nodes.

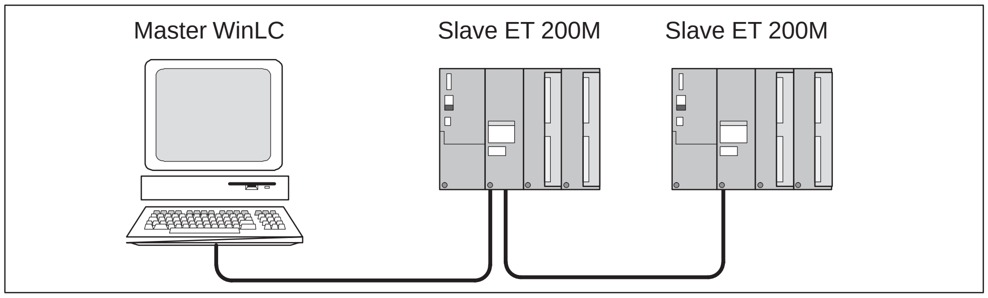


Figure 6-1 Sample PROFIBUS-DP network

Assigning the Node Addresses

You must assign a node address for each node on the network (from 0 to 125). Each address must be unique. You do not assign a node address to a repeater.

The default address for the DP master (WinLC) is 2. Typically, you would reserve address 0 for a programming device that is to be connected temporarily for maintenance or commissioning. Figure 6-2 shows a sample PROFIBUS-DP network with typical addresses for the nodes.

Depending on the type of device, a node address may be assigned using physical switches on the device or using a configuration tool. Consult your device documentation for the procedure required for your specific PROFIBUS–DP device.

Note

You are not required to assign consecutive addresses; however, performance is improved when the addresses are consecutive.

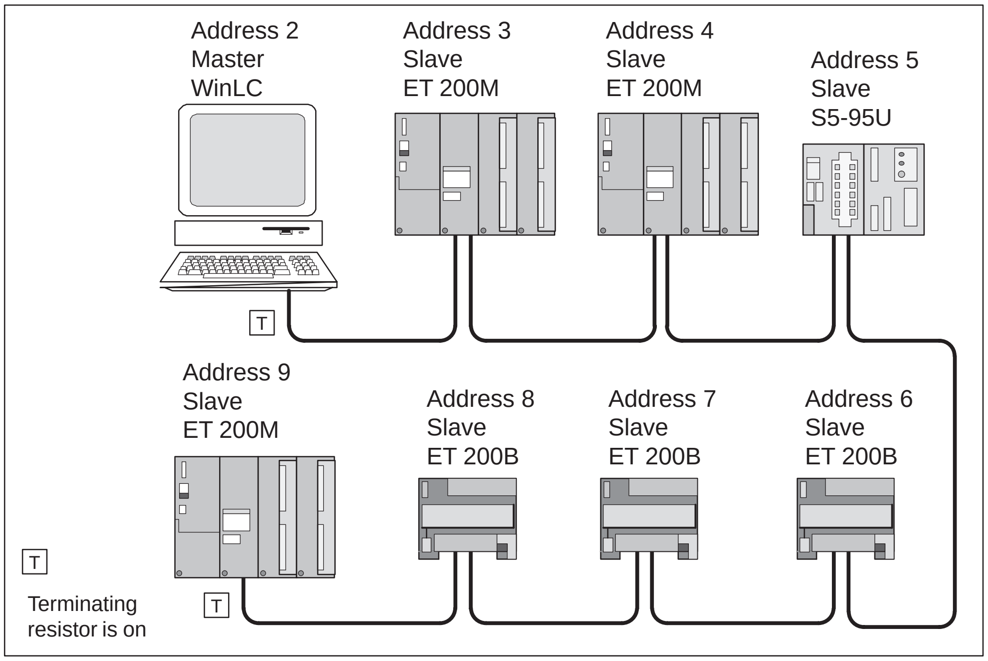


Figure 6-2 Typical Addresses for a PROFIBUS-DP Network

Installation Guidelines

Use the following guidelines for configuring and installing your DP network:

- Before connecting a node to the network, insure that its node address has been correctly set. Depending on the device, you may need to use the STEP 7 programming software to assign both the PROFIBUS node address or you may need to set the address using switches on the device. (You do not assign a node address to repeater.) Clearly label each node with its node address.
- Reserve node address 0 for a programming device that will be connected to the network on a temporary basis (such as to provide maintenance or commissioning).
- Turn on the terminating resistor for the nodes on either end of a network segment. For all other nodes, ensure that the terminating resistor is turned off.
- To connect more than 32 nodes on the network, use repeaters to create additional segments for the network.

  You can connect multiple segments to create a network; however, the signal path between any two nodes of the network must not cross more than ten segments. While each segment can consist of up to 32 nodes, the total network cannot exceed 126 nodes.
- When adding a new node to the network, turn off the power to the node before connecting it to the network.
- Use spur lines to connect any programming device or operator panel that will be used for startup or maintenance. If your network communicates at 3 Mbaud or more, use a special high-speed cable.
- All of the nodes in a segment must be connected in a linear construction (in a row from one node to the next). If your network communicates at 3 Mbaud or more, use special high-speed bus connectors.

Guidelines for Using Repeaters

Use the following guidelines for networks that utilize repeaters:

- Use repeater modules to connect segments of the network or to extend the cable length between nodes, or to connect non-grounded bus segments with grounded bus segments.
- You do not assign a node address to the repeater.
- Each repeater on the network segment counts as a node (against the maximum number of 32 nodes for the segment) and reduces the number of available nodes that you can connect to the segment.

  Even though the repeater counts as one of the 32 nodes (maximum) that can be physically connected on a segment, the repeater is not included as one of the 126 addressable nodes (maximum) for the PROFIBUS network.

6.2 Determining the Physical Layout of the Network

The requirements of the distributed I/O determine the physical layout of the network. These factors include the distance between stations, the number of nodes, and the different types of nodes being used.

Determining the Maximum Length of a Segment

Each segment of the PROFIBUS-DP network is limited to a maximum distance (or cable length). As shown in Figure 6-3, the maximum length of cable for a segment is measured between the nodes that have terminating resistors.

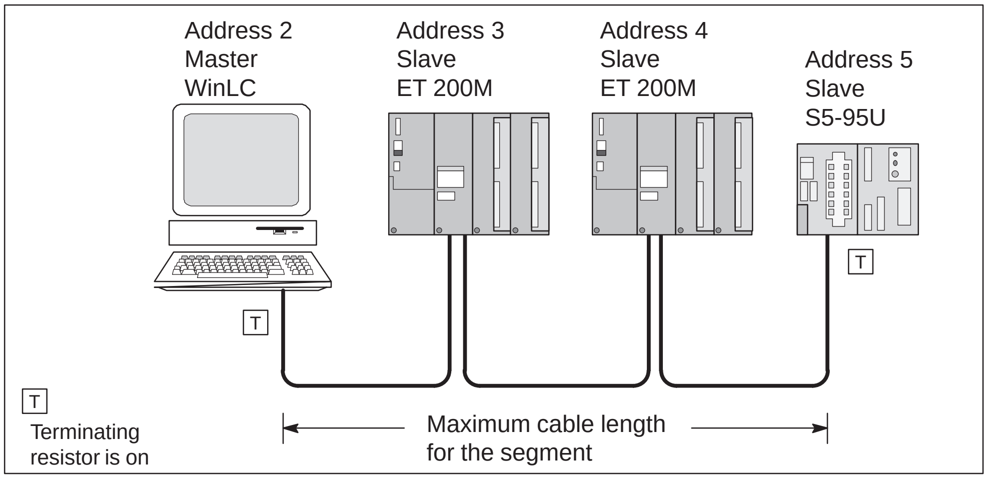


Figure 6-3 Maximum Cable Length for a Segment

The maximum distance for a segment is determined by the baud rate of the communication. Table 6-1 lists the maximum length of a segment for the baud rates which are supported by PROFIBUS-DP. For example, if the segment shown in Figure 6-3 uses 187.5 Kbaud, the maximum cable length is 1000 m (3280 ft.).

Table 6-1 Baud Rate and Maximum Cable Length

Baud Rate	Maximum Cable Length
9.6 Kbaud 93.75 Kbaud 19.2 Kbaud	1200 m (3936 ft.) with an isolated interface
187.5 Kbaud	1000 m (3280 ft.) with an isolated interface
500 Kbaud	400 m (1312 ft.)
1.5 Mbaud	200 m (656 ft.)
3 Mbaud 12 Mbaud 6 Mbaud	100 m (328 ft.)

Using Repeaters to Extend the Maximum Length

To communicate over distances which are greater than the maximum cable length allowed in Table 6-1, you must use a repeater with the network. In the example shown in Figure 6-4, two repeaters connect two nodes at a distance of 1100 m (3608 ft.), which is longer than the 1000 m (3280 ft.) maximum.

The maximum distance between two repeaters corresponds to the maximum cable lengths of a segment (see Table 6-1). You can connect up to nine repeaters in series.

The repeater counts as a node, but is not assigned a PROFIBUS address.

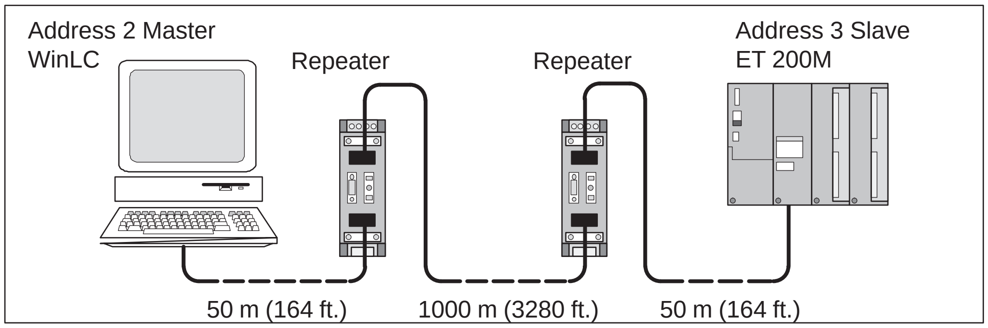


Figure 6-4 Using Repeaters to Increase the PROFIBUS Cable Length

Using Spur Lines

You can use a spur line to connect a node to a terminal block or other connector, instead of connecting the node directly to the PROFIBUS-DP cable. Table 6-2 lists the maximum lengths for spur lines. For networks that communicate at 3 Mbaud or greater, use a special, high-speed cable (order number 6ES7 901–4BD00–0XA0). This cable also allows you to connect more than one programming device.

Table 6-2 Length of Spur Line per Segment

Baud Rate	Maximum Length per Segment	Number of Nodes for Spur Line Length of:	
		1.5 m (4.9 ft.)	3 m (9.8 ft.)
9.6 Kbaud to 93.75 Kbaud	96 m (314.8 ft.)	32	32
187.5 Kbaud	75 m (246.0 ft.)	32	25
500 Kbaud	30 m (98.4 ft.)	20	10
1.5 Mbaud	10 m (32.8 ft.)	6	3
3 Mbaud to 12 Mbaud	Use the high-speed cable (order number 6ES7 901–4BD00–0XA0). (Spur lines are not allowed.)		

6.3 Assigning the Addresses for the Distributed I/O

You specify the PROFIBUS–DP configuration using the hardware configuration tool of the STEP 7 programming software. This includes specifying the node and diagnostic addresses for each node of the network as well as the logical addresses for the I/O data presented to WinLC by each node's modules. The PROFIBUS–DP configuration must be downloaded to WinLC before you attempt to operate the PROFIBUS network.

As stated previously, each node of the DP network has a unique node address. This address is used by the DP–Master to communicate with its DP–slaves; however, the user program generally does not reference a node's data using the node address. Instead, as part of the configuration process, the user assigns a "diagnostic" address to the node and a "logical" address range to the Input and Output data areas of the node's modules. This is accomplished using the hardware configuration tool of the STEP 7 programming software. Table 6-3 provides an overview of the addresses which may be assigned to the distributed I/O for WinLC.

Table 6-3 Address Ranges for the Distributed I/O

Address Areas	Size	
Process image areas	512 bytes:	IB0 to IB511 QB0 to QB511
	1024 bytes:	IB0 to IB1023 QB0 to QB1023
Total amount for the distributed I/O (accessed by Load and Transfer instructions)	16 Kbytes:	PIB0 to PIB16383 PQB0 to PQB16383
Total amount for consistent data (accessed by SFC14 and SFC15)	Up to 16 Kbytes (16384 bytes) for inputs and 16 Kbytes (16384 bytes) for outputs	
Maximum size per SFC14 or SFC15	240 bytes	
Maximum Input/Output data for one node	Up to122 bytes	

Specifying the Node Addresses

When you place a DP device into the WinLC configuration using the hardware configuration function of STEP 7, you are prompted to enter the device's node address. This address identifies the node to the DP master system. STEP 7 defaults WinLC's node address to the value 2. As slaves (I/O modules/racks) are placed on the DP "wire", STEP 7 displays a default node address. You may change this address, if required.

Specifying the Logical I/O Address

During the configuration of WinLC, the I/O data of each module of the DP–Network is allocated a logical address in the input and/or output area. You use these addresses to access the module’s input or output data, respectively. Additionally, a module’s base (lowest logical) address is used by WinLC to report module events to the user’s program.

Table 6-4 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. See Figure 6-5. I/O data may be assessed 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 240 bytes), use SFC14 (DPRD_DAT) and SFC15 (DPWR_DAT). SFC14 and SFC15 always access the module’s peripheral image.

Table 6-4 Accessing the Distributed I/O

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 the following instructions: • 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 in units other than 1 byte, 2 bytes and 4 bytes (up to 240 bytes)	Use the following SFCs: • SFC14 copies up to 240 bytes from a module’s inputs to the I, Q, M, D or L area. • SFC15 writes up to 240 bytes from the I, Q, M, D, or L area to a module’s outputs.

As shown in Figure 6-5, the user 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,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, PQ) or when using SFC14 or SFC15.

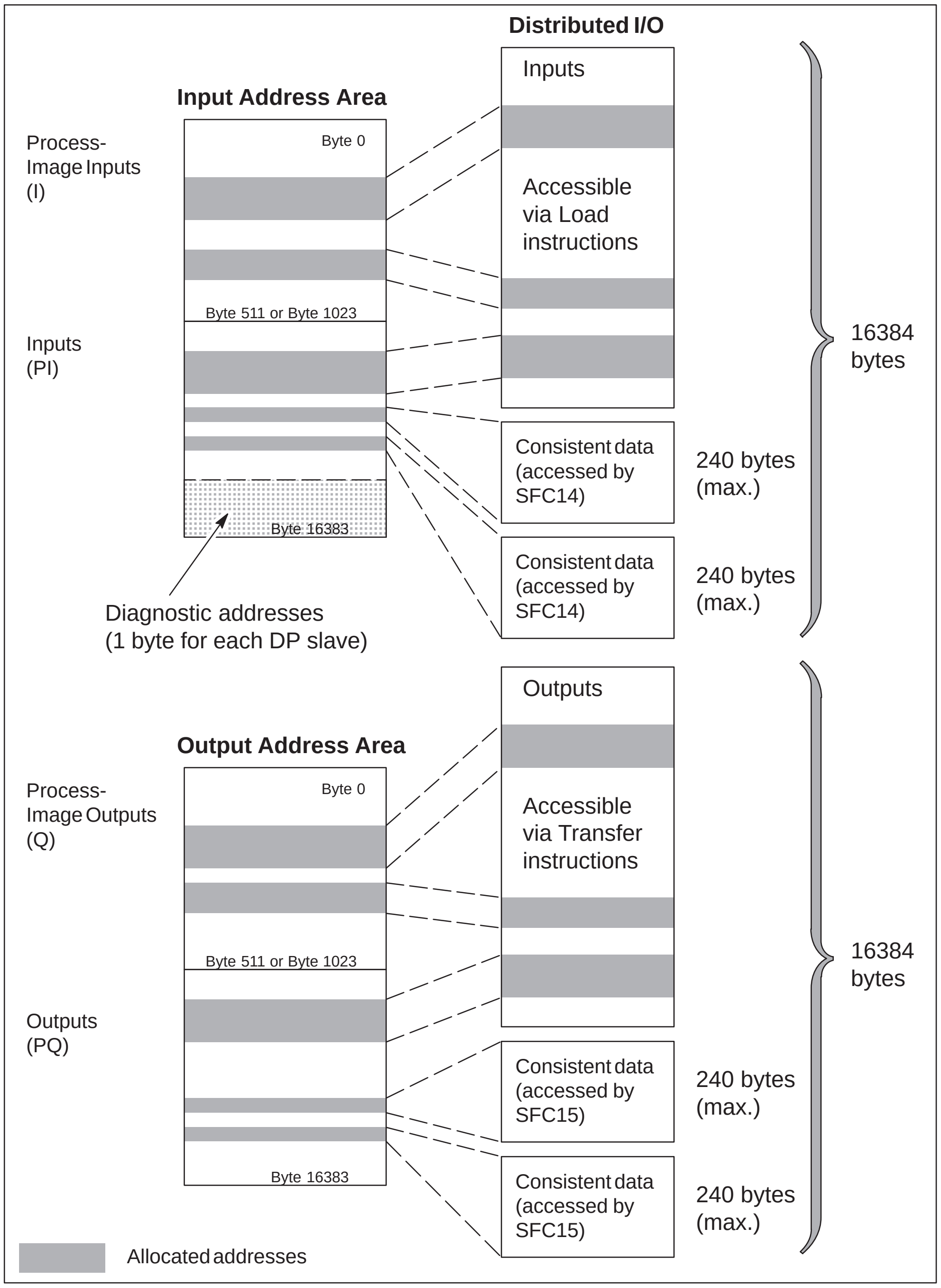


Figure 6-5 Accessing the Distributed I/O

SFC14 and SFC15 can access blocks of data up to 240 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.

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 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 Manual*.

For information about SFC14 (DPRD_DAT) and SFC15 (DPWR_DAT), see the *System Software for S7-300 and S7-400 System and Standard Functions Reference Manual*.

Specifying the Diagnostic Addresses

During the configuration of WinLC, each node of the DP Network is allocated a diagnostic address in the peripheral input (PI) area. You use the diagnostic address in parameters for the SFCs that access the node’s diagnostic data (for example, the LADDR parameter of SFC13). Additionally, this address is used by WinLC to report node state changes (in OB86) to the user’s program.

Note

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

As you use the STEP 7 hardware configuration tools to configure WinLC and the PROFIBUS-DP network, these diagnostic addresses are assigned above the process-image input (I) memory area. See Figure 6-6. If you do not enter a specific address, STEP 7 allocates IB16383 for the first DP slave, PIB16382 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, the *STEP 7 User Manual*, and the *SIMATIC NET PROFIBUS User Manual*.

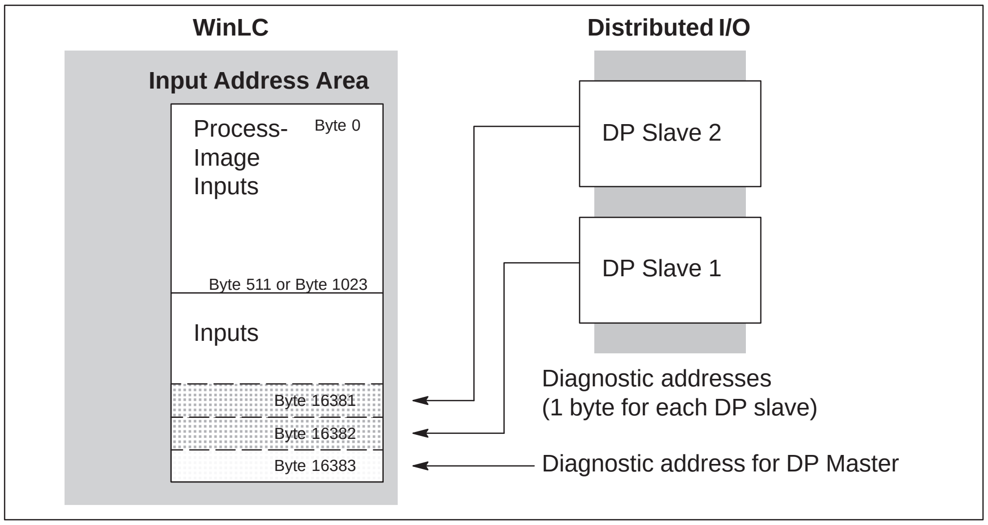


Figure 6-6 Diagnostic Addresses for the Distributed I/O

Troubleshooting for Network Problems

WinLC provides powerful features to help you diagnose DP network problems. First, the CPU panel provides two status indicators (EXT1 and BUSF1) that can be used to diagnose problems with the PROFIBUS-DP network. Table 6-5 describes the activity of the EXTF and BUSF1 indicators to help you determine the type of problem and a possible solution.

Table 6-5 Using the EXTF and BUSF1 Indicator

EXTF	BUSF1	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.
Off	Off	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	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 may 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.

6.4 Starting the PROFIBUS-DP Network

There are two elements of the hardware configuration in the STEP 7 programming software that affect the PROFIBUS-DP network:

- The “Startup” tab of the STEP 7 Hardware Configuration configures the startup parameters for the WinLC controller. These parameters are stored in the System Data container, which is downloaded with the user program from STEP 7 to the controller.
- The STEP 7 Hardware Configuration also maintains the PROFIBUS-DP network configuration. This information is stored in the System Data container, which is downloaded with the user program from STEP 7 to the controller.

For more information about configuring the PROFIBUS-DP and the startup parameters for the controller, see Section 5.5 (and Table 5-5), the online help for the STEP 7 programming software and the *STEP 7 User Manual*.

Turning On the Network

After configuring the PROFIBUS-DP network, use the following procedure to turn on the network:

1. With the controller in STOP mode, download the configuration of the PROFIBUS-DP network. You can download just the System Data (hardware configuration) or you can download all of the blocks of the user program.
2. Turn on all of the DP slaves on the PROFIBUS-DP network.
3. Wait until the EXTF and BUSF1 LEDs are both off.
4. Switch the operating mode of the controller from STOP to RUN.

Responding to Diagnostic Events

If an error is detected by the controller, the error condition is logged in the diagnostics 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, to delay, or to 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. For information about using these OBs, refer to the *System Software for S7-300 and S7-400 System and Standard Functions Reference Manual*.

You can also use SFC13 (DPNRM_DG) to read the diagnostic information from a DP slave. For information about SFC13, refer to the *System Software for S7-300 and S7-400 System and Standard Functions Reference Manual*.

System Status List (SZL) A

The information in the system status list (SZL) is stored as a set of sublists. Each sublist has a two-word header that 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.

SFC51 (RDSYSST) accesses the entries in the system status list. For more information about the system status list, see the *System Software for S7-300 and S7-400 System and Standard Functions Reference Manual*.

Table A-1 provides an overview of the SZL sublists, sorted according to the SZL-ID. You use the SZL-ID and index (as hexadecimal numbers: 16#) to access the records stored in the sublist.

Table A-1 Sublists of the System Status List (SZL) for the WinLC

SZL-ID (hexadecimal)	**Sublist**	**Index (hexadecimal)**	**Record Contents**
	SZL-ID		Information on all available SZL-IDs
0000	All available SZL-IDs	0131	Indices for SZL-ID 0131
0300	Lists the available indices	0132	Indices for SZL-ID 0132
		0222	Indices for SZL-ID 0222
	CPU identification		WinLC type and version number
0011	All records of the sublist	0001	Identification of the module
0111	One record of the sublist	0007	Identification of the firmware
0F11	Header information only		
	CPU features		
0112	Only those records of a group of features	0100	Time system in WinLC
		0200	System response
		0300	Language description of WinLC
0F12	Header information only		
0013	**User memory areas**	Work memory, integrated Load memory, plugged-in Load memory, maximum number of plug-in Load memories, and size of backup memory	
0014	**Operating system areas**	Process-image input area (bytes), process-image output area (bytes), bit memory (bytes), timers, counters, size of the I/O address area, and total local data area for WinLC (bytes)	

Table A-1 Sublists of the System Status List (SZL) for the WinLC, continued

SZL-ID (hexadecimal)	Sublist	Index (hexadecimal)	Record Contents	
	Block types			
0015	All records of the sublist		OBs (number and size) DBs (number and size) SDBs (number and size) FCs (number and size) FBs (number and size)	
0115	One record, depending on the index	0800	OBs (number and size)	
	State of the module LEDs			
0019	Status of all LEDs			
0119	Status of each LED	0002	INTF	Internal failure
		0003	EXTF	External failure
		0004	RUN	Run
		0005	STOP	Stop
		0006	FRCE	Force
		0007	CRST	Complete restart
		0008	BAF	Battery failure
		000B	BUSF1	Bus fault
0F19	Header information only			
	Interrupt/error assignment (via number of assigned OBs)			
0021	All possible interrupts			
0F21	Header information only			
	Interrupt status			
0222	Record for specified interrupt	0001 0050	Event that started OB1 Event that started OB80	
	Priority class			
0023	Records for all priority classes	0000	Priority of possible OBs	
0123	Records for a specific priority class			
0223	Records for all configured priority classes			
0F23	Header information only			
	Operating status of the CPU			
0124	Last executed operating status transition			
0424	Current operating status			
0524	Specified operating status	4520 5000 5010 5020 5030	Defective status STOP status Startup status RUN status HOLD status	

Table A-1 Sublists of the System Status List (SZL) for the WinLC, continued

SZL-ID (hexadecimal)	Sublist	Index (hexadecimal)	Record Contents
0131	**Communications performance parameters** of the communications type specified	0001 0002 0003 0004 0005 0009	Number of connections and baud rates Test and startup parameters Operator interface parameters Object management system (operating system function) Diagnostics functions and diagnostics entries Number of run-time meters
0132	**Communications status information** of the communications type specified	0001 0002 0004 0008 0009 000A	Number and type of connections Number of test jobs set up Protection levels of the WinLC Time system, correction factor, run-time meter, date and time of day Baud rate (set by means of the MPI) Baud rate (set by means of the S7-300 backplane bus)
	Diagnostic Station List		
0033	All entries		
	Start-up events		
0782	Start-up events of all OBs of a priority class before processing	Priority class	Event ID, priority class, and OB number
	Module status information		
0A91	Status information of all DP subsystems and DP masters		
0C91	Status information of a module	Start address	Features and parameters of the module
0D91	Status information for the specified station	*xxyy*	All the modules of station *yy* in the PROFIBUS-DP network *xx*
0F91	Header information only		
	Status information of the nodes in a DP network		Status information for the nodes connected to a PROFIBUS-DP network
0092	Target status of nodes in a subnetwork	0000	
0292	Actual status of the nodes in a subnetwork	Subnetwork ID	
0692	DP slaves indicating failure of one or more modules		
0F92	Header information only		

Table A-1 Sublists of the System Status List (SZL) for the WinLC, continued

SZL-ID (hexadecimal)	**Sublist**	**Index (hexadecimal)**	**Record Contents**
	Diagnostics buffer	Event information (dependent on the event)	
00A0	All entries (event information)		
01A0	Specified number of entries		
0FA0	Header information only		
	Module diagnostics		Module-dependent diagnostics information
00B1	Data record 0 of the module diagnostics information	Start address	Starting address for the specific module
00B3	Data record 0 of the module diagnostics information (Complete module-dependent diagnostics)	Rack and slot number	Rack and slot number of the specific module
00B4	DP–norm diagnostics of a DP slave		

Instruction List

B

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. In addition to these system blocks, you can use the other S7 blocks to create the user 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.

An OB can also contain 65,570 bytes.

For more information about OBs, SFCs, and SFBs, see the *System Software for S7-300 and S7-400 System and Standard Functions Reference Manual*.

B.1 Technical Data

Order Number

WinLC is a component of the WinAC Basis package: 6ES7 671–0CC00–0YX0

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 which is running WinLC
- Load memory (RAM): equal to the amount of memory (physical plus virtual) allowed by the operating system (Windows NT)
- 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 either 512 bytes or 1024 bytes. These memory areas can be accessed directly by the instructions in the user 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

Table B-1 Performance Characteristics and Technical Specifications of WinLC

WinLC	Description
Work memory	Limited to the amount of physical memory on the computer that is running WinLC
Load memory (RAM)	Limited to the amount of memory (physical plus virtual) allowed by the operating system (Windows NT)
Accumulators	4 (ACCU 1 to ACCU 4)
Local data	16 Kbytes per priority class
Clock	Real-time system clock, based on the hardware clock of the computer

Table B-1 Performance Characteristics and Technical Specifications of WinLC, continued

WinLC	Description
Digital I/O (digital and analog)	16384 bytes (inputs) and 16384 bytes (outputs)
Process image I/O (user configurable) • Inputs • Outputs	512 bytes (inputs) and 512 bytes (outputs) or 1024 bytes (inputs) and 1024 bytes (outputs) • I 0.0 to I 511.7 or I 0.0 to I 1023.7 • Q0.0 to Q511.7 or Q0.0 to Q1023.7
Memory bits • Retentive range (configurable) • Preset as retentive	2 Kbytes • MB0 to MB255 • 16 bytes (MB0 to MB15)
Counters • Retentive range (configurable) • Preset as retentive	512 • C0 to C63 • 8 (C0 to C7)
Timers (only updated in OB1) • Retentive range (configurable) • Preset as retentive	512 • T0 to T127 • None
Clock memory Bits of the clock memory byte toggle at specific times and are accessible from the user program.	8 bits of clock memory (1 byte) 8 frequencies within 1 byte of M memory: address is configurable
Number of blocks supported • OB • SFB • SFC • Maximum number of asynchronous SFCs • Address ranges for logic blocks: – FB – FC – DB • Total number of blocks that can be downloaded to WinLC	 15 (see Table B-2) 7 (see Table B-8) 52 (see Table B-6) 20 FB0 to FB65535 FC0 to FC65535 DB1 to DB65535 (DB0 is reserved) 4,000
Nesting depth	24 per OB, with 2 asynchronous OBs (OB121 and OB122) per priority class
PROFIBUS-DP interface • DP address area • Number of DP slaves supported • Baud rate • Baud rate search (as a DP slave) • Transfer memory (as a DP slave) • Maximum distance	 • 16384 bytes (inputs) and 16384 bytes (outputs) • 125 (CP5412 A2 is limited to 64 slaves) • 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) • Not applicable • Not applicable • Dependent on the baud rate (see Table 6-1)

B.2 Organization Blocks (OBs) Supported

OBs are the interface between the operating system of WinLC and the user program. Table B-2 lists the OBs which are supported. WinLC executes OBs according to the priority class.

Table B-2 Organization Blocks (OBs) Supported

OB	Description	Priority Class
OB1	Main program cycle	1 (lowest)
OB10	Time-of-day interrupt	2
OB20	Time-delay interrupt	3 to 6
OB35, OB36	Cyclic interrupt	7 to 15
OB40	Hardware interrupt	16 to 23
OB80	Time error	26
OB82	Diagnostic interrupt	24 to 26 (or 28)[1]
OB83	Module remove/insert interrupt	24 to 26 (or 28)[1]
OB85	Priority class error	24 to 26 (or 28)[1]
OB86	Rack failure	24 to 26 (or 28)[1]
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	

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

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

Table B-3 shows OBs for the main program cycle and cold and warm restarts. WinLC provides OB1 (main program cycle) for continuously executing the user program. On the transition from STOP mode to RUN mode (or RUN-P 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 box displayed by the CPU panel. After OB100 (or OB102) has been successfully executed, WinLC executes OB1.

Table B-3 OBs for the Main Program Cycle, Cold Restart, and Warm Restart

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

Interrupt OBs

WinLC provides a variety of OBs that interrupt the execution of OB1. Table B-4 lists the different interrupt OBs which 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 user program (or other OB) and executes the interrupting OB. You can change the priority class for the interrupt OBs (see Table B-2).

Table B-4 Interrupt OBs

Interrupts		Start Event	Default Priority	
Time-of-Day Interrupt	OB10	1111$_H$ (OB10)	2	Low
Time-Delay Interrupt Range: 1 ms to 60000 ms	OB20	1121$_H$ (OB20)	3	↓
Cyclic Interrupt Range: 1 ms to 60000 ms Recommended: > 10 ms	OB35 OB36	1136$_H$ 1137$_H$	12 13	
Hardware interrupt	OB40	1141$_H$ (channel 1)	16	High

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, or OB40 is missing and OB85 has not been downloaded, WinLC changes operating mode (from RUN to STOP).
- WinLC remains in RUN mode if OB35 or OB36 is missing or cannot be executed at the specified time.

Note

You can configure OB35 and OB36 to be executed as frequently as every 10 ms. If you schedule an OB to be executed every 10 ms, make certain that the program can be executed within the time frame and also that your WinLC application can process the OB within the allotted time.

Error OBs

As shown in Table B-5, 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 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.

Table B-5 Error OBs

Error or Fault		Start Event	Default Priority
Time-out error	OB80	3501_H, 3502_H, 3505_H, 3507_H	26
Diagnostic Interrupt	OB82	3842_H, 3942_H	26
Insert/remove module interrupt	OB83	3861_H, 3863_H, 3864_H, 3961_H, 3865_H	26
Priority class error: • 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	$35A1_H$, $35A3_H$, $39B1_H$, $39B2_H$,	26
Distributed I/O failure: a node in the PROFIBUS-DP subnetwork has failed or been restored.	OB86	$38C4_H$, $39C4_H$, $38C5_H$, $39C5_H$, $38C7_H$, $38C8_H$,	26 (or 28)
Programming error (For example: the user program attempts to address a timer that does not exist.)	OB121	2521_H, 2522_H, 2523_H, 2524_H, 2525_H, 2526_H, 2527_H, 2528_H, 2529_H, 2530_H, 2531_H, 2532_H, 2533_H, 2534_H, 2535_H, $253A_H$; $253C_H$, $253E_H$	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_H, 2943_H	

B.3 System Functions (SFCs) Supported

WinLC provides SFCs, which are logic blocks that perform basic tasks. Table B-6 lists the SFCs which are supported. The user program calls the SFC and passes the required parameters; the SFC performs its task and returns the result.

WinLC allows a maximum of 20 asynchronous SFCs to be running. The following asynchronous SFCs are supported: SFC11, SFC13, SFC51 (index B1, B3), SFC55, SFC56, SFC57, SFC58, and SFC59.

Note

An asynchronous SFC is an SFC that has a “Busy” output parameter.

Table B-6 System Functions (SFCs) Supported

SFC	Name	Description	Execution Time	
SFC0	SET_CLK	Sets the system clock	48.46μs[1]	25.55 μs[2]
SFC1	READ_CLK	Reads the system clock	15.00 μs[1]	7.79 μs[2]
SFC2	SET_RTM	Sets the run-time meter	10.39 μs[1]	5.18 μs[2]
SFC3	CTRL_RTM	Starts or stops the run-time meter	10.73 μs[1]	5.50 μs[2]
SFC4	READ_RTM	Reads the run-time meter	7.55 μs[1]	3.88 μs[2]
SFC5	GADR_LGC	Queries the logical address of a channel	16.82 μs[1]	8.43 μs[2]
SFC6	RD_SINFO	Reads the start information of an OB	17.27 μs[1]	8.76 μs[2]
SFC11	DPSYNC_FR	Synchronize groups of DP slaves	10.30 μs[1]	5.20μs[2]
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	32.00 μs[1]	29.63 μs[2]
SFC14	DPRD_DAT	Reads the consistent data from a DP slave	29.48 μs[1]	14.45 μs[2]
SFC15	DPWR_DAT	Writes the consistent data to a DP slave	29.60 μs[1]	14.32 μs[2]
SFC17	ALARM_SQ	Generates an acknowledgeable block-related message	46.95 μs[1]	25.18 μs[2]
SFC18	ALARM_S	Generates an unacknowledgeable block-related message	53.36 μs[1]	27.31 μs[2]
SFC19	ALARM_SC	Queries the status for the last message (SFC17 or SFC18)	11.94 μs[1]	6.04 μs[2]
SFC20	BLKMOVB	Copies variables	37.67 μs[1]	18.85 μs[2]
SFC21	FILL	Initializes a memory area 1 word 50 words 100 words	38.10 μs[1] 60.90 μs 83.26 μs	19.03 μs[2] 31.02 μs 42.51 μs
SFC22	CREAT_DB	Creates a data block	65.91 μs[1]	35.02 μs[2]
SFC23	DEL_DB	Deletes a data block	21.97 μs[1]	11.67 μs[2]
SFC24	TEST_DB	Provides information about a data block	11.45 μs[1]	5.60 μs[2]
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	1831.73 μs[1]	1282.40 μs[2]
SFC27	UPDAT_PO	Updates the process-image output table	1831.63 μs[1]	1286.20 μs[2]

Table B-6 System Functions (SFCs) Supported, continued

SFC	Name	Description	Execution Time	
SFC28	SET_TINT	Sets the time-of-day interrupt (OB10)	20.64 µs[1]	10.16 µs[2]
SFC29	CAN_TINT	Cancels the time-of-day interrupt (OB10)	11.01 µs[1]	5.26 µs[2]
SFC30	ACT_TINT	Activates the time-of-day interrupt (OB10)	9.49 µs[1]	4.94 µs[2]
SFC31	QRY_TINT	Queries the time-of-day interrupt (OB10)	11.09 µs[1]	5.56 µs[2]
SFC32	SRT_DINT	Starts the time-delay interrupt (OB20)	29.62 µs[1]	14.29 µs[2]
SFC33	CAN_DINT	Cancels the time-delay interrupt (OB20)	12.80 µs[1]	5.87 µs[2]
SFC34	QRY_DINT	Queries the time-delay interrupt (OB20)	10.91 µs[1]	5.53 µs[2]
SFC36	MSK_FLT	Mask synchronous errors	10.09 µs[1]	5.50 µs[2]
SFC37	DMSK_FLT	Unmask synchronous errors	9.70 µs[1]	4.92 µs[2]
SFC38	READ_ERR	Read the error register	9.66 µs[1]	4.86 µs[2]
SFC39	DIS_IRT	Disables the processing of all new interrupts	9.80 µs[1]	5.17 µs[2]
SFC40	EN_IRT	Enables the processing of new interrupts	10.03 µs[1]	4.93 µs[2]
SFC41	DIS_AIRT	Disables the processing of new interrupts with higher priority than the current OB	7.16 µs[1]	3.58 µs[2]
SFC42	EN_AIRT	Enables the processing of new interrupts with higher priority than the current OB	15.13 µs[1]	7.77 µs[2]
SFC43	RE_TRIGR	Retriggers the watchdog timer (monitoring the cycle time)	2551.39 µs[1]	2407.46 µs[2]
SFC44	REPL_VAL	Transfers a value to ACCU1 (accumulator 1)	14.20 µs[1]	10.75 µs[2]
SFC46	STP	Changes the operating mode to STOP	Not applicable	
SFC47	WAIT	Delays the execution of the user program	1951.82 µs[1]	1951.41 µs[2]
SFC49	LGC_GADR	Queries the module slot belonging to a logical address	13.24 µs[1]	6.99 µs[2]
SFC50	RD_LGADR	Queries all of the logical addresses of a module	44.73 µs[1]	22.06 µs[2]
SFC51	RDSYSST	Reads all or part of a system status list	49.16 µs[1]	22.42 µs[2]
SFC52	WR_UMSG	Writes a user element to the diagnostics buffer	57.22 µs[1]	29.23 µs[2]
SFC54	RD_PARM	Read the defined parameter	27.99 µs[1]	14.17 µs[2]
SFC55	WR_PARM	Write the defined parameter	46.22 µs[1]	24.03 µs[2]
SFC56	WR_DPARM	Write the default parameter	28.45 µs[1]	13.63 µs[2]
SFC57	PARM_MOD	Assign the parameters to a module	28.08 µs[1]	13.58 µs[2]
SFC58	WR_REC	Write a data record	46.11 µs[1]	24.01 µs[2]
SFC59	RD_REC	Read a data record	46.10 µs[1]	23.37 µs[2]
SFC64	TIME_TCK	Reads the time from the system clock	8.00 µs[1]	4.21 µs[2]
SFC79	SET	Set a range of outputs	11.41µs[1]	5.83 µs[2]
SFC80	RESET	Reset a range of outputs	11.43 µs[1]	5.83 µs[2]

1 The execution times were measured on a Hewlett-Packard HP Vectra XU computer (with dual Pentium processor) with 96 Mbytes of RAM running at 200 MHz, and a CP5412 DP card. Actual execution times will vary, depending on your system.

2 The execution times were measured on a Dell Dual 400 computer (with dual Pentium processor) with 128 Mbytes of RAM running at 400 MHz, and a CP5412 DP card. Actual execution times will vary, depending on your system.

B.4 Execution Times of DP Instructions

The execution times listed in Table B-7 are dependent on the DP card used.

Table B-7 Execution Times of DP Instructions

SFC	Name	Description	CP5412–A2[1]	CP5613[1]
SFC11	DPSYNC_FR	Synchronize groups of DP slaves	10.30 μs	10.045 μs
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	32.00 μs	28.03333 μs
SFC14	DPRD_DAT	Reads the consistent data from a DP slave	29.48 μs	28.34567 μs
SFC15	DPWR_DAT	Writes the consistent data to a DP slave	29.60 μs	27.896 μs
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	1831.73 μs	125.2333 μs
SFC27	UPDAT_PO	Updates the process-image output table	1831.63 μs	92.46667 μs

1 The execution times were measured on a Hewlett-Packard HP Vectra XU computer (with dual Pentium processor) with 96 Mbytes of RAM running at 200 MHz. Actual execution times will vary, depending on your system.

B.5 System Function Blocks (SFBs) Supported

WinLC provides SFBs, which are logic blocks similar to SFCs. Table B-8 lists the SFBs which are supported. When the user program calls an SFB, a data block (DB) must also be assigned.

Table B-8 System Function Blocks (SFBs) Supported

SFB	Name	Description	Execution Time	
SFB0	CTU	Provides a “count up” timer	18.00 μs[1]	9.08 μs[2]
SFB1	CTD	Provides a “count down” timer	18.87 μs[1]	9.11 μs[2]
SFB2	CTUD	Provides a “count up/down” timer	21.89 μs[1]	11.04 μs[2]
SFB3	TP	Generates a pulse	19.26 μs[1]	9.41 μs[2]
SFB4	TON	Generates an on-delay timer	18.36 μs[1]	9.41 μs[2]
SFB5	TOF	Generates an off-delay timer	18.93 μs[1]	9.54 μs[2]
SFB32	DRUM	Implements a sequencer	61.54 μs[1]	29.96 μs[2]

1 The execution times were measured on a Hewlett-Packard HP Vectra XU computer (with dual Pentium processor) with 96 Mbytes of RAM running at 200 MHz. Actual execution times will vary, depending on your system.

2 The execution times were measured on a Dell Dual 400 computer (with dual Pentium processor) with 128 Mbytes of RAM running at 400 MHz. Actual execution times will vary, depending on your system.

B.6 Execution Times of Instructions

The execution times listed in Table B-9 (execution times for math operations) and Table B-10 (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.

Table B-9 Execution Times of Math Operations (μs)

Math Operation	**Integer**		**Real**		**Double Word**	
Addition (+)	0.09μs[1]	0.06 μs[2]	0.15 μs[1]	0.13 μs[2]	0.08 μs[1]	0.07 μs[2]
Subtraction (–)	0.08 μs[1]	0.06 μs[2]	0.15 μs[1]	0.13 μs[2]	0.12 μs[1]	0.12 μs[2]
Multiplication (*)	0.01 μs[1]	0.06 μs[2]	0.15 μs[1]	0.13 μs[2]	0.28 μs[1]	0.18 μs[2]
Division (/)	0.47 μs[1]	0.26 μs[2]	0.30 μs[1]	0.19 μs[2]	0.28 μs[1]	0.18 μs[2]

1 Execution times were measured on a Hewlett-Packard HP Vectra XU computer (with dual Pentium processor) with 96 Mbytes of RAM running at 200 MHz. Actual execution times may vary, depending on your system.

2 Execution times were measured on a Dell Dual 400 computer (with dual Pentium processor) with 128 Mbytes of RAM running at 400 MHz. Actual execution times may vary, depending on your system.

Table B-10 Execution Times (µs) per Instruction

Instructions			Execution Time (200 MHz)[1]		Execution Time (400 MHz)[2]	
			Direct addressing	**Indirect addressing**	**Direct addressing**	**Indirect addressing**
Boolean operations: A, AN O, ON, X, XN	Memory areas:	I	0.55 µs	0.69 µs	0.26 µs	0.30 µs
		M	0.49 µs	0.68 µs	0.23 µs	0.31 µs
		L	0.64 µs	0.68 µs	0.32 µs	0.32 µs
		DB	0.62 µs	0.75 µs	0.29 µs	0.36 µs
		T	1.53 µs	1.85 µs	0.75 µs	0.91 µs
		C	0.47 µs	0.72 µs	0.23 µs	0.36 µs
Boolean operations (on the accumulator): ==I, <>I, >I, <I, >=I, <=I			0.38 µs		0.21 µs	
Operations on the bits of the status word: A==0, A<>0, A>0, A<0, A>=0, A<=0			0.43 µs		0.24 µs	
Transitional contacts:	Edge Positive	FP	0.75 µs		0.39 µs	
	Edge Negative	FN	0.75 µs		0.39 µs	
Set/Reset operations	Set	S	0.55 µs	0.65 µs	0.31 µs	0.32 µs
(bit operands)	Reset	R	0.52 µs	0.82 µs	0.30 µs	0.42 µs
RLO Operations	Negate RLO	NOT	0.36 µs		0.19 µs	
	Set RLO	SET	0.34 µs		0.19 µs	
	Clear RLO	CLR	0.29 µs		0.16 µs	
	Save RLO	SAVE	0.34 µs		0.19 µs	
Operations on Timers	Pulse timer	SP	1.59 µs	1.98 µs	0.81 µs	1.01 µs
	Reset (timer)	R	0.36 µs	0.65 µs	0.19 µs	0.32 µs
	Extended pulse timer	SE	1.59 µs	1.98 µs	0.81 µs	0.97 µs
	On-delay timer	SD	1.59 µs	1.95 µs	0.81 µs	0.97 µs
	Retentive on-delay timer	SS	1.59 µs	1.95 µs	0.81 µs	0.96 µs
	Off-delay timer	SF	1.72 µs	2.08 µs	0.86 µs	1.01 µs
Miscellaneous:	Open DB	OPN	1.30µs		0.67 µs	
	Load	L	0.48 µs		0.31 µs	
	Transfer	T	0.55 µs		0.26 µs	

1 The execution times were measured on a Hewlett-Packard HP Vectra XU computer (with dual Pentium processor) with 96 Mbytes of RAM running at 200 MHz. Actual execution times may vary, depending on your system.

2 The execution times were measured on a Dell Dual 400 computer (with dual Pentium processor) with 128 Mbytes of RAM running at 400 MHz. Actual execution times may vary, depending on your system.

B.7 Allowing for Scan Time “Jitter”

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” (where the scan cycle varies from the configured minimum scan time—see Section 4.6). For example, different video cards and IDE hard drives can affect the specified performance of WinLC.

Examples of Jitter in the Scan Cycle

To show typical jitter, the performance of WinLC was tested on two computers:

- Dell 400 computer (single 400 MHz Pentium processor) with 128 Mbytes of RAM. This computer also has a SCSI hard drive, a Colorgraphic Predator dual monitor graphics card, and a 3Com Ethernet card.
- Hewlett-Packard HP Vectra XU computer (dual 200 MHz Pentium processor) with 96 Mbytes of RAM. This computer also has a SCSI hard drive, an ATI graphics card, and a 3Com Ethernet card.
- Dell Dual 400 computer (dual 400 MHz Pentium processor) with 128 Mbytes of RAM. This computer also has a SCSI hard drive, a Colorgraphic Predator dual monitor graphics card, and a 3Com Ethernet card.

The performance test consisted of running a variety of software concurrently under the following conditions:

- WinLC executed a test program with 3000 Boolean operations, reading and writing data to I/O modules over a PROFIBUS-DP network consisting of three nodes (ET200M). In addition, WinLC performed calculations to determine the distribution of performance data.
- WinLC was configured to run with the following parameters:
  - Minimum scan = 25 ms
  - Priority = 31 “Real-time/Critical” (single Pentium: 26 “Real-time/High”)
  - Minimum sleep time = 0 ms (single Pentium: 1 ms)

  (For information about configuring scan time, sleep time and priority, see Section 4.6.)
- While WinLC executed the test program, the computer also executed the following:
  - STEP 7 monitored WinLC operations (updating 512 double-words of data in the Monitor and Modify Variables application).
  - Two graphics-intensive test programs ran in infinite loops.
  - Microsoft Excel 97 used a SIMATIC Data control to read a block of 1024 double-words of data.
  - Various Microsoft software applications (such as Word 97, Excel 97, Internet Explorer 4, and Outlook 98) were opened, used, and closed.

During the testing, Windows NT Task Manager measured the load on the computer:

- For the single Pentium Dell computer, the CPU utilization ranged between 99% and 100%. The average execution time of the WinLC program was 11 ms, with an average sleep time of 11 ms to 14 ms.
- For the dual Pentium (200 MHz) HP computer, the CPU utilization ranged between 99% and 100%. The average execution time of the WinLC program was 23 ms, with an average sleep time of 2 ms to 5 ms.
- For the dual Pentium (400 MHz) Dell computer, the CPU utilization ranged between 85% and 90%. The average execution time of the WinLC program was 11 ms, with an average sleep time of 14 ms.

Figure B-1 shows the jitter in the scan time as measured on the two different computers. Actual performance may vary, depending on your system.

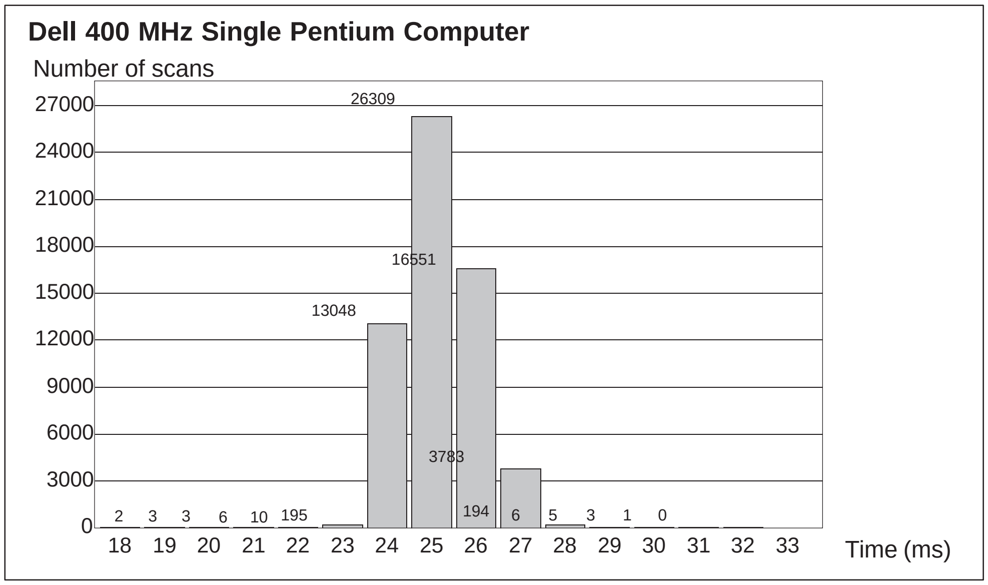


Figure B-1 Sample Jitter for the Dell 400 MHz Single Pentium Computer

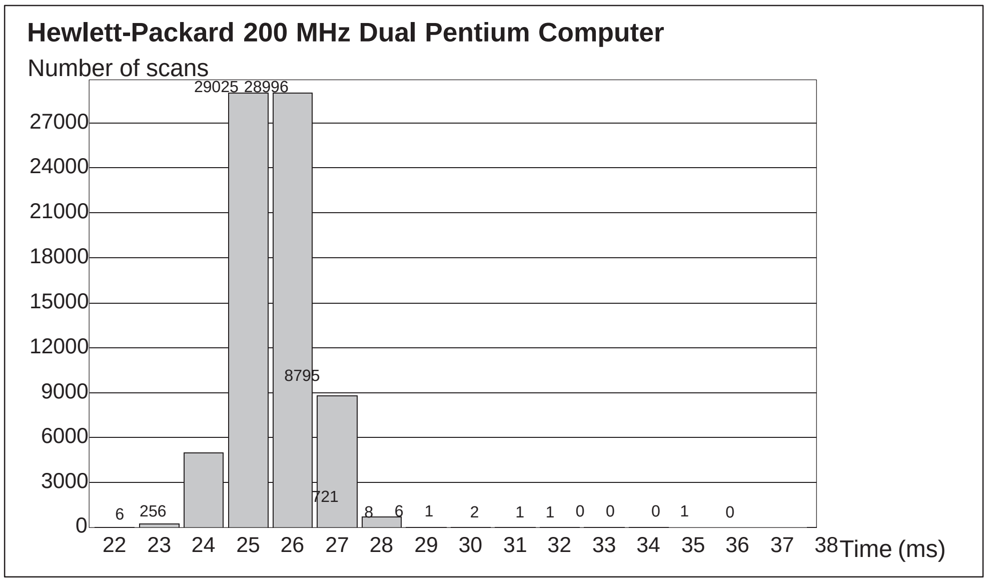


Figure B-2 Sample Jitter for the Hewlett–Packard 200 MHz Dual Pentium Computer

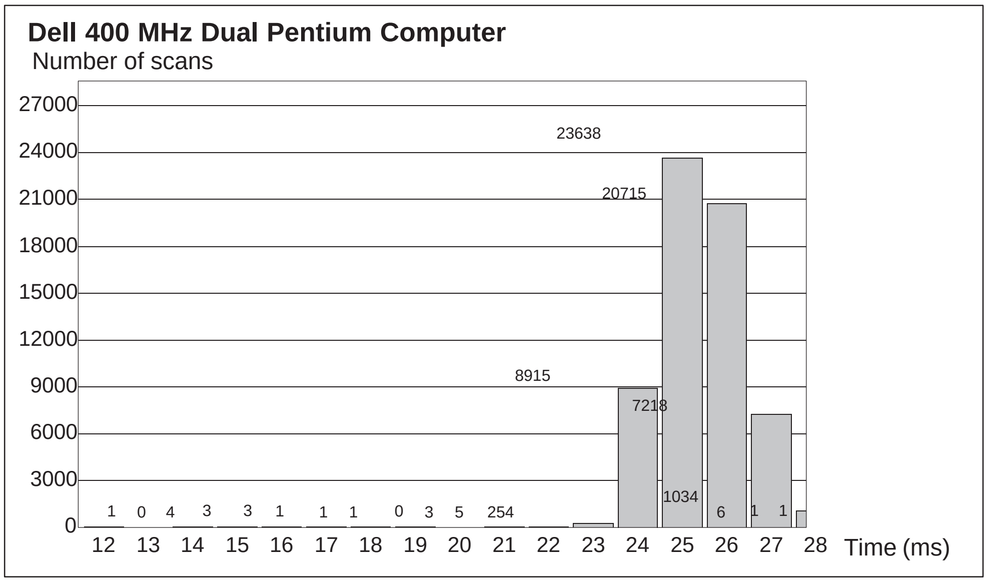


Figure B-3 Sample Jitter for the Dell 400 MHz Dual Pentium Computer

Panel Control

C

Chapter Overview

The Panel is also available as an ActiveX component for use in SIMATIC Computing. Panel Control permits access from SIMATIC Computing's SoftContainer or from any ActiveX container.

The Panel Control provides access to the operating modes of either WinLC (WinAC Basis) or a slot PLC (WinAC Pro). You can change the operating mode from STOP to RUN or RUN-P, or you can use the MRES button to reset the memory areas of the controller.

C.1 Accessing the Controller with the Panel Control

The Panel control corresponds to the faceplate of the S7 CPU modules. As shown in Figure C-1, the control contains buttons for changing the operating mode of the controller, a button for resetting the memory area, and status indicators.

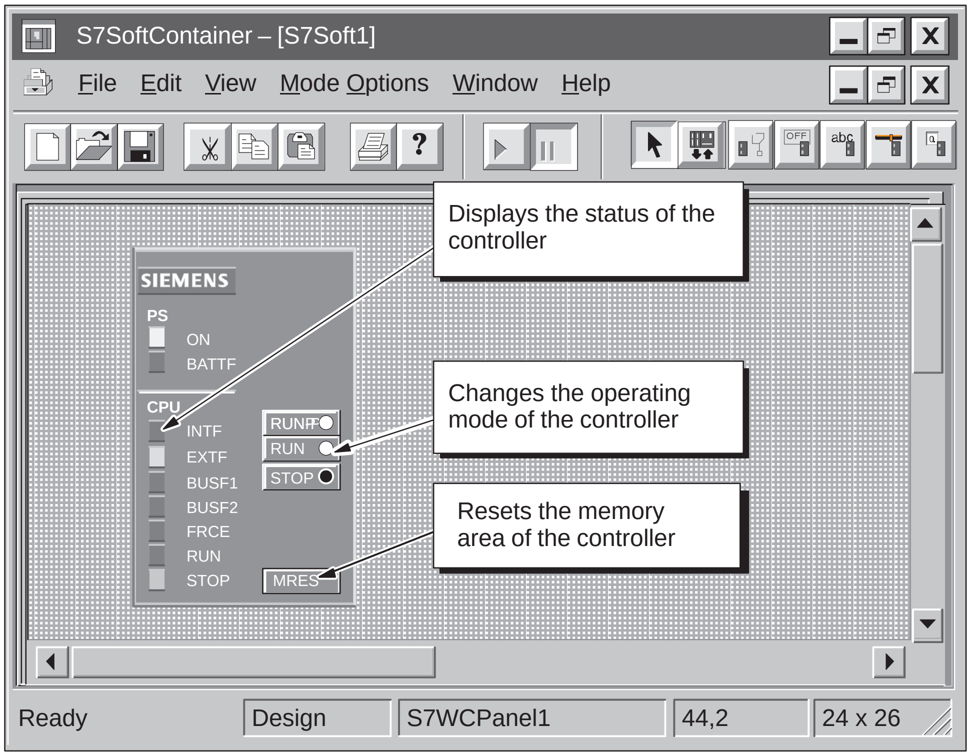

S7SoftContainer – [S7Soft1]
File
Edit
View
Mode Options
Window
Help
Displays the status of the controller
Changes the operating mode of the controller
Resets the memory area of the controller
SIEMENS
PS
ON
BATTF
CPU
INTF
EXTF
BUSF1
BUSF2
FRCE
RUN
STOP
RUNP
RUN
STOP
MRES
Ready
Design
S7WCPanel1
44,2
24 x 26

Figure C-1 Buttons and Indicators on the Panel Control

Warning

When you change the operating mode selection of the Panel control, you are changing the operating mode of the controller in your actual process. If you select the MRES button, a memory reset is issued to the controller.

Resetting or changing the mode of the controller interrupts process operation. If equipment is not in a safe state, interrupting the process could result in death or serious injury to personnel, and/or damage to equipment.

Do not allow anyone to change the mode of the controller or issue a reset unless you have ensured that your equipment is in a safe state. Always install a physical emergency stop circuit for your machine or process.

Selecting the Operating Mode

The RUN, RUNP, and STOP buttons on the Panel control correspond to the different operating modes of the controller, and are shown in Table C-1:

- In STOP mode, the controller is not executing the user program. To download a program that includes SDBs, you must place the controller in STOP mode. On the transition to STOP mode, the outputs go to a safe state.
- In RUN mode, the controller executes the user program. You cannot download any new user program or logic blocks when the controller is in RUN mode.
- In RUN-P mode, the controller executes the user program. You can download new programs or logic blocks.

Clicking on the button places the controller into the selected operating mode. The status indicators on the Panel control show whether the controller is in RUN (or RUN-P) mode or in STOP mode.

To allow an external source, such as the STEP 7 programming software, to change the operating mode of the controller, select either RUN or RUN-P mode. If the operating mode is changed by the external software, the selected button on the Panel control does not change, but the status indicators reflect the actual operating mode of the controller.

Table C-1 Buttons for Changing the Operating Modes of the Controller

Mode	Description
RUNP	The controller executes the user program. When the controller is in RUN-P mode (RUN-PROGRAM mode), you can: • Upload a program from the controller to your computer or programming device • Download a program to the controller • Download individual blocks to the controller
RUN	The controller executes the user program. You can upload a program from the controller to your computer or programming device, but you cannot download a program to the controller.
STOP	The controller does not execute the user program. When the controller is in STOP mode, you can: • Upload a program from the controller to your computer or programming device • Download a program to the controller

Using the Status Indicators

The status indicators (BUSF1, BUSF2, INTF, EXTF, PS, BATTF, FRCE, RUN, and STOP) show basic information about the controller, such as the current operating mode or the presence of an error condition. Table C-2 describes the different status indicators for the CPU panel of the controller. You cannot change the status of the controller by clicking on the status indicators.

If there is a break point in the user program, both the RUN and STOP indicators turn on while the break point is active: the RUN indicator flashes, and the STOP indicator is on.

During a restart, both the RUN and STOP indicators are turned on: the RUN indicator flashes, and the STOP indicator is on during the restart; when the STOP indicator turns off, the outputs are enabled.

If all of the status indicators are flashing, the controller is defective.

Table C-2 Status Indicators

Indicator	Description
ON	Power supply. Always on for WinLC.
BATTF	Battery fault. Always off for WinLC.
INTF	This indicator lights up (solid) to show error conditions within the controller, such as programming errors, firmware errors, arithmetic errors and timer errors.
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	These indicators light up (either solid or flashing) to identify fault conditions in the communication with the distributed I/O. See Table 6-5. Since WinLC supports only 1 PROFIBUS-DP network, BUSF1 is the only active indicator; BUSF2 is not applicable for WinLC.
FRCE	This indicator lights up (solid) to show that a force request is active. Not applicable for WinLC.
RUN STOP	Lights up (solid) to show the operating mode (RUN or STOP) When RUN is flashing and STOP is lighted (solid): • The controller is executing a restart. • The user program has reached a break point.
All status indicators are flashing	When all of the status indicators are flashing, WinLC has encountered an error condition that cannot be fixed by resetting the memory (MRES). To recover from this condition, you must perform the following tasks: 1. Shut down the WinLC controller. 2. Restart the WinLC controller. 3. Reset the memory (MRES). If WinLC is running as a service, you must use the Windows NT control panel to shut down and restart the WinLC controller.

Using the MRES Button to Reset the Memory Areas

The Panel control provides a MRES button for resetting the memory areas to the default values and deleting the user program. Clicking the MRES button places the controller into STOP mode and performs the following tasks:

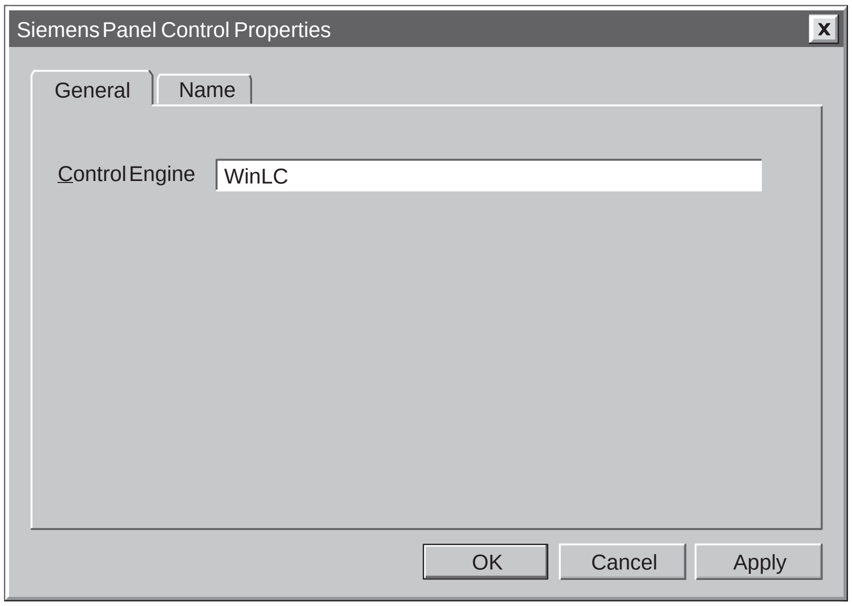
1. The controller deletes the entire user program, including the data blocks (DBs).
2. The controller resets the memory areas (I, Q, M, T, and C).

After the memory has been reset, the diagnostics buffer remains intact, as does the MPI address.

C.2 Selecting the Control Engine for the Panel Control

When using the ActiveX Panel Control, you must specify the Control Engine to which to connect. The Panel does not connect to hardware PLCs or across networks. Figure C-2 shows the Properties dialog box for the Panel control. You enter the name of the controller in the “Control Engine” property field:

- **WinLC** (for WinLC of WinAC Basis)
- **CPU 416–2 DP ISA** (for the CPU 416-2 DP ISA of WinAC Pro)
- other slot PLC



Control Engine: WinLC

Figure C-2 Panel Control Properties (General Tab)

Note

If you are using a third-party container that allows you to view the other properties for the Panel control, do not modify these properties or the values assigned to them.

C.3 Sample Programs Using the Panel Control

You can write programs to initiate actions based the status of the Panel control. The following sample programs provide examples of how you can write programs that use the Panel control.

Changing the Operating Mode of the Controller

Your program can change the operating mode (RUN, RUN-P, STOP) of the controller. Table C-3 provides sample subroutines that performs these tasks when you click on a command button in the VB form.

- If the subroutine ConnectToCPU is called, the Panel control connects to a specific controller
- If the subroutine SetToRun is called, the operating mode of the controller changes to RUN mode.
- If the subroutine SetToRunP is called, the operating mode of the controller changes to RUN-P mode.
- If the subroutine SetToStop is called, the operating mode of the controller changes to STOP mode.

Table C-3 Connecting to a Controller and Changing the Operating States

Visual Basic Code

```
Private Sub ConnectToCPU
   S7Panel1.ConnectCPU = True   ’Connect the Panel to the Selected Control Engine
End Sub
```

```
Private Sub SetToRun
   S7Panel1.ModeCtrl = RUN_Switch        ’Place WinLC in Run Mode
End Sub
```

```
Private Sub SetToRunP
   S7Panel1.ModeCtrl = RUNP_Switch       ’Place WinLC in Run–P Mode
End Sub
```

```
Private Sub SetToStop
   S7Panel1.ModeCtrl = STOP_Switch       ’Place WinLC in Stop Mode
End Sub
```

Configuring the Security State of the Panel Control

You can design a custom application that uses a Panel control, but allows the security for the application to determine whether a user can operate the Panel control. Since your application will have its own password or other security checking, you do not require any additional security checking to be performed by the Panel control.

The sample subroutines listed in Table C-4 provide sample code for accomplishing the following tasks:

- To bypass the security provided by the Panel control, you set the SecurityState property of the Panel control to **`App_Does_Security`**. The Panel control now relies on the application to determine whether the user has permission to make changes in the controller.

  In this example, the SecurityState property is set to this value when the form for the application loads.

- To ensure that a user must have permission from the application before allowing any changes to be made with the Panel control, you set the SwitchOK property of the Panel control to False. The button on the Panel control will now respond to user requests only if the application changes the state of the SwitchOK property.

  In this example, the SwitchOK property is set to False when the form for the application loads.

- To enable the user to make changes to the controller with the Panel control, your application sets the SwitchOK property of the Panel control to True.

  When the PerformSecurityCheck subroutine determines that the user has permission to make changes with the Panel control, the subroutine sets the SwitchOK property of the Panel control to True. Until the SwitchOK property is set to True, the Panel control does not make the change requested by the user.

Using this sample code, any time that a user requests that the Panel control perform some task, the Panel control determines whether the user has been given permission by the application to make the requested change. For example, when a user clicks on the “RUN” button of the Panel control to change the controller from STOP mode to Run mode, the Panel control checks the state of the SwitchOK property before changing the operating mode of the controller.

Table C-4 Configuring the Security State for the Panel Control

Visual Basic Code

```
’This sample application uses a Boolean parameter (AppPasswordValid)
’to allow changes to be made with the Panel control

   Dim AppPasswordValid As Boolean ’User is (or is not) allowed to make changes
```

```
Private Sub Form_Load()
   ’This section connects the Panel control to the controller (WinLC) and initializes
   ’the properties of the Panel control

   ’Set the Control Engine String for the controller
      S7Panel.ControlEngine = WinLC

   ’Connect the Panel control to WinLC
      S7Panel.ConnectCPU = True

   ’Initialize the SwitchOK property to False. This prevents any changes to be made
   ’until the application performs the security check
      S7Panel.SwitchOK = False

   ’Set the security state to have the application perform the security check
      S7Panel.SecurityState = App_Does_Security

End Sub
```

```
Private Sub PerformSecurityCheck()
   ’This subroutine provides the security checking for the application.
   ’
      ’The code that checks the security for the application goes here...
      ’If the user has permission to make changes, AppPasswordValid is set to True
      ’Otherwise, AppPasswordValid is set to False

   ’State of AppPasswordValid determines whether the Panel control responds to user
      S7Panel.SwitchOK = AppPasswordValid

End Sub
```

Responding to Changes in the State of the LEDs on the Panel Control

Table C-5 provides a sample subroutine that reads the state of the LED for RUN mode and determines the color of the LED and if the LED is turned on or is flashing. The constants which are declared for the subroutine are the masks for the values in the LED properties: CpuBusf1, CpuBusf2, CpuExtF, CpuFrce, CpuIntF, CpuRun, and CpuStop.

Table C-5 Responding to Changes in the LEDs of the Panel Control

Visual Basic Code

```
Private Sub S7Panel_UpdateState()
      ’These constants are the masks for the LED properties:
   Const LED_GREEN = &H2
   Const LED_3SEC = &H100
   Const LED_ON = &H200
   Const LED_05HZ = &H300
   Const LED_20HZ = &H400
      ’For this example, RunLedColorTxt and RunLedStateTxt are text fields:
      ’RunLedColorTxt displays a message about the color of the RUN mode LED
      ’RunLedStateTxt displays a message about the state (on or flashing)
      ’ of the RUN mode LED

   If S7Panel.CpuRun = 0 Then
      RunLedColorTxt.Caption = “Color of the RUN mode LED is gray”
      RunLedStateTxt.Caption = “RunLED is Off”
   End If

   If ((S7Panel.CpuRun And LED_GREEN) = LED_GREEN) Then
      RunLedColorTxt.Caption = “Color of the RUN mode LED is green”
   End If

   If ((S7Panel.CpuRun And LED_ON) = LED_ON) Then
      RunLedColorTxt.Caption = “RUN mode LED is turned on (and is not flashing)”
   End If

   If ((S7Panel.CpuRun And LED_3SEC) = LED_3SEC) Then
      RunLedColorTxt.Caption = “RUN Mode LED flashes for 3 seconds”
   End If

   If ((S7Panel.CpuRun And LED_05SEC) = LED_05HZ) Then
      RunLedColorTxt.Caption = “RUN Mode LED flashes at 5 Hz intervals”
   End If

   If ((S7Panel.CpuRun And LED_20SEC) = LED_20HZ) Then
      RunLedColorTxt.Caption = “RUN Mode LED flashes at 20 Hz intervals”
   End If

End Sub
```

C.4 Evaluating the LEDs of the Panel Control

The Panel control has the following LED properties:

- CpuBusf1
- CpuBusf2
- CpuExtF
- CpuFrce
- CpuIntF
- CpuRun
- CpuStop

You use the constants (hexadecimal values) listed in Table C-6 to evaluate the states of the LEDs on the Panel control. These masks determine the status of the individual LED property.

Table C-6 Masks for the LEDs of the Panel Control

Mask (Hexadecimal Value)	Description
1	LED color = orange
2	LED color = green
3	LED color = red
100	LED flashes for 3 seconds
200	LED is on (solid, not flashing)
300	LED flashes at a frequency of 5 Hz
400	LED flashes at a frequency of 20 Hz

C.5 Properties and Methods of the Panel Control

ActiveFilePath Property

Applies to: Panel

This read-only property provides the pathname to the control engine (controller).

Syntax:

[*value* **=]** *object***.ActiveFilePath**

The ActiveFilePath property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A String expression that evaluates to the name of the controller.

AutoStart Property

Applies to: Panel

This property allows you to select the “autostart” feature for WinLC. This property is valid only for WinLC. For information about the autostart feature, refer to the WinLC documentation.

Syntax:

*object***.AutoStart [=** *value***]**

The AutoStart property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that specifies whether the autostart feature is enabled for *object*.

The settings for *value* are:

Setting	Description
True	The autostart feature of WinLC is enabled.
False	(default) The autostart feature of the WinLC is disabled.

CheckPW Property

Applies to: Panel

This property determines whether the password entered was correct. If the password entered matches the password stored in the control engine, the control executes the requested action.

Syntax:

object*.CheckPW [=** *value]**

The CheckPW property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A integer that determines whether *object* performs the requested action.

The settings for *value* are:

Setting	Description
0 – Check_Wait	(default) The control engine is verifying the password.
1 – Check_Good	The password entered was correct and the action is allowed.
2 – Check_Bad	The password entered was incorrect and the action is not allowed.

ConnectCPU Property

Applies to: Panel

This property establishes a connection to or disconnects from the S7 controller (WinLC or any of the slot PLCs listed in section C.2).

Syntax:

*object***.ConnectCPU [=** *value***]**

The ConnectCPU method has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that specifies whether *object* establishes a connection to the S7 control engine.

The settings for *value* are:

Setting	Description
True	*Object* connects to the S7 controller.
False	(default) *Object* disconnects from the S7 controller.

ControlEngine Property

Applies to: Panel

This property stores the pathname or identification of the control engine connected to the control.

Syntax:

object.**`ControlEngine [=`** *value***`]`**

The ControlEngine property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A String that specifies the pathname or identification of the control engine to be accessed by *object*.

CpuBusf1, CpuBusf2 Properties

Applies to: Panel

This read-only property determines the state of the communication indicators (BUSF1 and BUSF2) on the control. BUSF1 shows the status of the distributed (remote) I/O for the control engine; if a second network is supported by the control engine, BUSF2 shows the status of the second network.

Syntax:

`[`*value* **`=]`** *object*.**`CpuBusf1`**
`[`*value* **`=]`** *object*.**`CpuBusf2`**

The CpuBusf1 and CpuBusf2 properties have these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression that specifies the status of the bus fault (BUSF1 or BUSF2) on *object*.

Settings for *value* are shown in Table C-6.

CpuExtF Property

Applies to: Panel

This read-only property determines the state of the External Fault indicator on the control. External faults are errors that are detected outside the CPU module of the control engine, such as broken wiring for the local I/O.

Syntax:

[*value* **=]** *object***.CpuExtF**

The CpuExtF property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression that specifies the status of the EXTF indicator on *object*.

The settings for *value* are shown in Table C-6:

CpuFrce Property

Applies to: Panel

This read-only property determines the state of the FRCE indicator on the control. The FRCE indicator lights to signal that a user-generated Force request is in effect. (Using programming software such as STEP 7, the user can stipulate that the control engine set or force an input or output to a specific value.)

Syntax:

[*value* **=]** ***object*****.CpuFrce**

The CpuFrce property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression that specifies the status of the FRCE indicator on *object*.

The settings for *value* are shown in Table C-6.

CpuIntF Property

Applies to: Panel

This read-only property determines the state of the Internal Fault indicator on the control. Internal faults are errors that are detected within the CPU module of the control engine, such as programming errors that cause the control engine to go to STOP mode.

Syntax:

[*value* **=]** *object***.CpuIntF**

The CpuIntF property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression that specifies the status of the INTF indicator on *object*.

The settings for *value* are shown in Table C-6:

CpuRun Property

Applies to: Panel

This read-only property determines the state of the RUN mode indicator on the control.

Syntax:

[*value* **=]** *object***.CpuRun**

The CpuRun property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression that specifies the state of the RUN indicator.

The settings for *value* are shown in Table C-6:

CPURunning Property

Applies to: Panel

This read-only property indicates that the control engine is still running or in operation. The control queries the control engine, and if the control engine responds, the property is set to True.

Syntax:

[*value* **=]** *object***.CpuRunning**

The CpuRunning property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that specifies whether the control engine is running and able to respond to the control.

The settings for *value* are:

Setting	Description
True	The control engine is running and has responded to the query by the control
False	(default) The control engine is not running or not responding.

CpuStop Property

Applies to: Panel

This read-only property determines the state of the STOP mode indicator on the control.

Syntax:

`[`*value* **`=]`** *object*.**`CpuStop`**

The CpuStop property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression that specifies the state of the STOP indicator.

The settings for *value* are shown in Table C-6:

FirmwareVersion Property

Applies to: Panel

This read-only property stores the revision level of the firmware in the control engine.

Syntax:

`[`*value* **`=]`** *object*.**`FirmwareVersion`**

The FirmwareVersion property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A String value that describes the revision level of the firmware for the control engine.

FmrSwitch Property

Applies to: Panel

This property restarts the backup battery of the slot PLC.

Syntax:

object.**`FmrSwitch [=`** *value***`]`**

The FmrSwitch property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that causes the control engine to restart the backup battery.

The settings for *value* are:

Setting	Description
True	The control engine performs a battery restart (FMR).
False	(default) No action is required.

HardwareVersion Property

Applies to: Panel

This read-only property stores the version (revision level) of the control engine hardware.

Syntax:

`[`*value* **`=]`** *object*.**`HardwareVersion`**

The HardwareVersion property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A String value that describes the hardware version for the control engine.

mlfb Property

Applies to: Panel

This read-only property stores the order number for the slot PLC.

Syntax:

[*value* **=]** *object***.mlfb**

The mlfb property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A String value that specifies the order number for the control engine.

ModeCtrl Property

Applies to: Panel

This property changes the operating mode of the control engine.

Syntax:

*object***.ModeCtrl [=** *value***]**

The ModeCtrl property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer that determines the new operating mode for the control engine.

The settings for *value* are:

Setting	Description
0	MRES (memory restart)
1	STOP mode
2	RUN mode
3	RUN-P mode

OnStateChanged Method

Applies to: Panel

This method is used internally by the control and must not be modified.

PSBattF Property

Applies to: Panel

This read-only property determines the state of the Battery Fault indicator on the control. This property is valid for the control engine. The BATTF indicator lights to alert the user to a battery fault condition.

Syntax:

[*value* **=]** *object***.PSBattF**

The PSBattF property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression that specifies the status of the BATTF indicator on *object*.

The settings for *value* are shown in Table C-6:

PSOn Property

Applies to: Panel

This read-only property determines the state of the power supply (ON) indicator on the control. The ON indicator shows the status of the power supply for the control engine.

Syntax:

[*value* **=]** *object***.PSOn**

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer expression that specifies the status of the PS indicator on *object*.

The settings for *value* are shown in Table C-6:

PwrSwitch Property

Applies to: Panel

This property indicates the on/off status of the control engine.

Syntax:

object.**`PwrSwitch [=`** *value***`]`**

The PwrSwitch property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that specifies whether the control engine is on or off.

The settings for *value* are:

Setting	Description
True	The control engine is on.
False	The control engine is off.

ResourceFile Property

Applies to: Panel

This read-only property determines the name of the DLL for the language-specific Strings displayed by the control.

Syntax:

object.**`ResourceFile [=`** *value***`]`**

The ResourceFile property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A String that determines the name for the language-specific DLL.

ResourcePath Property

Applies to: Panel

This read-only property contains the pathname of the language-specific DLL selected for the control.

Syntax:

object.**`ResourcePath [=`** *value***`]`**

The ResourcePath property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A String that determines the pathname for the language-specific DLL.

SecurityState Property

Applies to: Panel

This property determines the level of security in effect for the control:

- Panel control handles security checking.
- Disables the security checking by the control. Your application performs all of the security. (See also the SwitchOK property.)

Syntax:

object.**`SecurityState [=`** *value***`]`**

The SecurityState property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	An integer that determines the level of security for *object*.

The settings for *value* are:

Setting	Description
0	Panel control provides security checks.
1	The security checking performed by the control is disabled. Your application performs all management of security. (See also the SwitchOK property.)

SetPassword Property

Applies to: Panel

If set to True, this property executes the “Set Password” function for changing the password in the control engine.

Syntax:

object.**`SetPassword [=`** *value***`]`**

The SetPassword property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that determines whether to call the “Set Password” function.

The settings for *value* are:

Setting	Description
True	The control calls the “Set Password” function for changing the password in the control engine.
False	(default) No action.

ShowErrorBoxes Property

Applies to: Panel

This property specifies whether to display the default error boxes when there is a user-generated error. Every time an error occurs, an Error event will be generated. If the ShowErrorBoxes property is enabled (selected), a default error message box will be displayed.

All errors on connections are reported by the Connection Error event.

Syntax:

object.**`ShowErrorBoxes [=`** *value***`]`**

The ShowErrorBoxes property has these parts:

Part	Description
object	An object expression (identifier for the specific SIMATIC control) that evaluates to an object in the Applies To list.
value	A Boolean expression that specifies whether the control displays error boxes.

The settings for *value* are:

Setting	Description
True	(default) The control shows the default error boxes.
False	The error boxes are hidden.

SwitchOK Property

Applies to: Panel

If your application is handling the security (by disabling the security checking normally performed by the control), this property allows a requested action to be performed. When the SecurityState property is set to 3, the control waits until the SwitchOK property is set to True before performing any action requested by a user. If the SecurityState property is set to 4, this property must be set to “True” in order for any action to take place.

Syntax:

*object***.SwitchOK [=** *value***]**

The SwitchOK property has these parts:

Part	Description
object	An object expression that evaluates to an object in the Applies To list.
value	A Boolean expression that allows or disallows an action to be performed.

The settings for *value* are:

Setting	Description
True	The user has permission to affect the requested action. The control then performs the requested action.
False	(default) The control does not perform the requested action.

C.6 Events of the Panel Control

AlarmCondition Event

Applies to: Panel

This event occurs when the Panel control detects that the control engine has an error condition or has gone to STOP mode.

Syntax: **`AlarmCondition()`**

ConnectionError Event

Applies to: Panel

This event occurs when an error on a connection occurs. The ConnectionError event provides no parameters.

Syntax:

`ConnectionError()`

MouseDown Event

Applies to: Panel

This event occurs when a mouse button is pressed while the mouse cursor is over the control.

Syntax:

MouseDown(short *Button***, short** *Shift***, OLE_XPOS_PIXELS** *x***, _
OLE_YPOS_PIXELS** *y***)**

The MouseDown event has these parts:

Part	Description
Button	An integer that identifies the button that was pressed to cause the event The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
Shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
x,*y*	returns a number that specifies the current location of the mouse pointer

MouseMove Event

Applies to: Panel

This event occurs when the mouse cursor moves over the control.

Syntax:

MouseMove(short *Button*, **short** *Shift*, **OLE_XPOS_PIXELS** *x*, **_**
OLE_YPOS_PIXELS *y***)**

The MouseMove event has these parts:

Part	Description
Button	An integer that identifies the button that was pressed to cause the event The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
Shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
x,*y*	returns a number that specifies the current location of the mouse pointer

MouseUp Event

Applies to: Panel

This event occurs when a mouse button is released while the mouse cursor is over the control.

Syntax:

MouseUp(short *Button*, **short** *Shift*, **OLE_XPOS_PIXELS** *x*, **_**
OLE_YPOS_PIXELS *y***)**

The MouseUp event has these parts:

Part	Description
Button	An integer that identifies the button that was pressed to cause the event The button argument is a bit field with bits corresponding to the left button (bit 0), right button (bit 1), and middle button (bit 2). These bits correspond to the values 1, 2, and 4, respectively. Only one of the bits is set, indicating the button that caused the event.
Shift	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys when the button specified in the button argument is pressed or released A bit is set if the key is down. The shift argument is a bit field with the least-significant bits corresponding to the SHIFT key (bit 0), the CTRL key (bit 1), and the ALT key (bit 2). These bits correspond to the values 1, 2, and 4, respectively. The shift argument indicates the state of these keys. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys are pressed. For example, if both CTRL and ALT were pressed, the value of shift would be 6.
x,*y*	returns a number that specifies the current location of the mouse pointer

MResBttnSelected Event

Applies to: Panel

This event occurs when a user selects (clicks on) the memory reset (MRES) button on the Panel control. You can use this event for implementing external security for your process.

Syntax: **`MResBttnSelected()`**

RunBttnSelected Event

Applies to: Panel

This event occurs when a user selects (clicks on) the RUN mode button on the Panel control. You can use this event for implementing external security for your process.

Syntax: **`RunBttnSelected()`**

RunPBttnSelected Event

Applies to: Panel

This event occurs when a user selects (clicks on) the RUN-P mode button on the Panel control. You can use this event for implementing external security for your process.

Syntax: **`RunPBttnSelected()`**

StopBttnSelected Event

Applies to: Panel

This event occurs when a user selects (clicks on) the STOP mode button on the Panel control. You can use this event for implementing external security for your process.

Syntax: **`StopBttnSelected()`**

UpdateState Event

Applies to: Panel

This event occurs when the Panel control detects a change in the status of the control engine.

Syntax: **`UpdateState()`**

Index

To

SIEMENS ENERGY & AUTOMATION INC

ATTN: TECHNICAL COMMUNICATIONS M/S 519

3000 BILL GARLAND ROAD

PO BOX 1255

JOHNSON CITY TN USA 37605–1255

From

Name:	_ _
Job Title:	_ _
Company Name:	_ _
Street:	_ _
City and State:	_ _
Country:	_ _
Telephone:	_ _

Please check any industry that applies to you:

❐ Automotive	❐ Pharmaceutical
❐ Chemical	❐ Plastic
❐ Electrical Machinery	❐ Pulp and Paper
❐ Food	❐ Textiles
❐ Instrument and Control	❐ Transportation
❐ Non-electrical Machinery	❐ Other ____________________________
❐ Petrochemical	

Remarks Form

Your comments and recommendations will help us to improve the quality and usefulness of our publications. Please take the first available opportunity to fill out this questionnaire and return it to Siemens.

Please give each of the following questions your own personal mark within a range from 1 (very good) to 5 (very poor).

1.	Do the contents meet your requirements?	☐
2.	Is the information you need easy to find?	☐
3.	Is the text easy to understand?	☐
4.	Does the level of technical detail meet your requirements?	☐
5.	Please rate the quality of the graphics and tables.	☐

Additional comments: