SIEMENS

SIMATIC

Windows Logic Controller WinLC RTX 3.1

User Manual

Edition 12/2001

A5E00083518-02

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

Warning

indicates a potentially hazardous situation which, if not avoided, could result in death or serious injury.

Caution

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

Caution

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

Notice

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

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.

Preface

The Windows Logic Controller (WinLC RTX) v. 3.1 provides the functionality of a programmable logic controller (PLC) in a real-time, PC-based environment. WinLC RTX uses the VenturCom Real-time extensions (RTX) to Windows NT and 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 RTX.

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

Audience

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

Scope of the Manual

This manual describes the features and the operation of WinLC RTX.

Other Manuals

You can find information in the online help for STEP 7 and for WinLC RTX. 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 WinLC RTX. 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	WinLC RTX 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.

Title	Content
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 WinLC.
SIMATIC NET PROFIBUS User Manual	This manual provides information about PROFIBUSDP 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) 7346500
  - (609) 7343530
- E-mail:
  - ISBU.Hotline@sea.siemens.com
  - simatic.hotline@sea.siemens.com
- Internet:
  - http://www.sea.siemens.com/software
  - http://www1.ad.siemens.de/meta/support/html_76/support.htm
  - http://www4.ad.siemens.de/csinfo/livelink.exe?func=cslib.csinfo2&siteid=cs&lang=en

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://www1.ad.siemens.de/meta/index00.htm

For information about VenturCom Real-Time extensions (RTX):

- Internet: http://www.vci.com

Contents

Product Overview 1

WinLC RTX provides real-time process control from your computer. As part of the SIMATIC family of automation products, WinLC RTX 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 RTX controller a powerful solution for your automation needs.

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

1.1 Controlling Your Process with WinLC RTX

WinLC RTX uses the VenturCom Real-time extensions (RTX) to the Windows NT operating system to provide a real-time, computer-based solution for your automation projects. As shown in Figure 1-1, WinLC RTX connects a PC-based controller over a PROFIBUS 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 RTX:

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

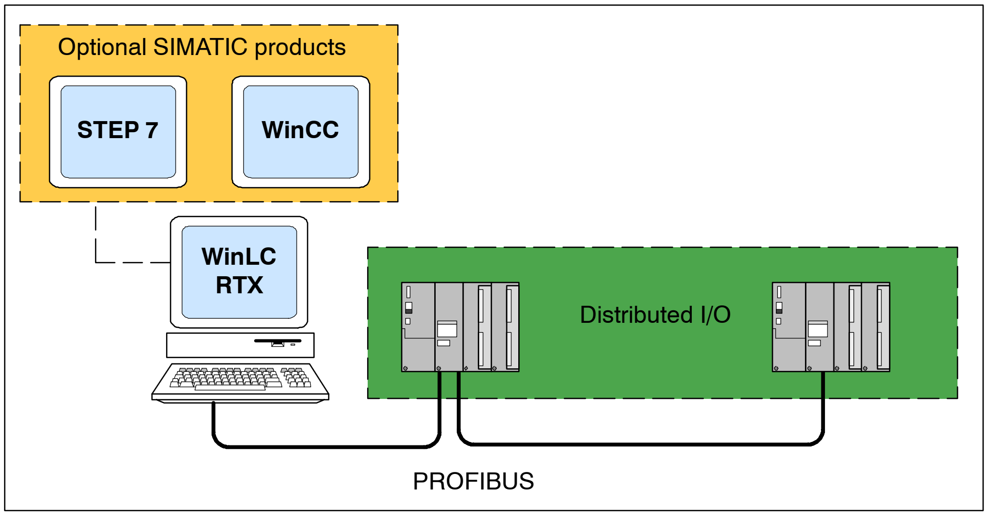


Figure 1-1 Components of WinLC RTX RT

Operational Features of WinLC RTX

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

System Requirements

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

- A personal computer (PC) with the following:
  - Pentium processor running at 400 MHz or faster (recommended)
  - 128 Mbytes RAM (recommended)
  - 512 Kbytes level 2 cache
  - Microsoft Windows NT version 4.0 (or higher), with service pack 6 (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 100 Mbytes of free space
- At least 1 Mbyte free space on drive C for the Setup program (Setup files are deleted when the installation is complete.)
- An installed CP 5613 card (Rev 3 or greater) connected to a PROFIBUS-DP network for distributed I/O communication.The CP5613 card must be installed in a slot that does not share an IRQ number with any Windows-controlled device.

1.2 Additional Features for WinLC RTX

WinLC RTX is a real-time PC-based controller that includes the following features:

- WinLC RTX 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 RTX can be used with earlier versions of STEP 7. For more information, see Section 4.2.
- WinLC RTX is uses the communications processor CP 5613 for communicating with the distributed I/O. Purchase the CP 5613 card card separately.
- If you are using STEP7 V5.0 SP3 or higher, configure WinLC RTX as a PC station.
- For improved deterministic behavior and isolation from NT failures, WinLC RTX executes the user program in the real-time subsystem.
- With WinLC RTX 3.1, you can operate the DP Master in constant bus cycle time (equidistant) mode.
- SFC 82, SFC 83, and SFC 84 allow you to create, copy and write data to Load memory.

1.3 How WinLC RTX Operates if Windows NT Crashes

WinLC RTX supports OB84 (CPU Hardware Fault), which allows you to initiate the shutdown of your process in case Windows NT detects an unrecoverable fault or STOP error while WinLC RTX is running. See Section 5.4 for more information.

1.4 Storing the Date

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

1.5 Differences from the Windows NT Version of WinLC

Some of the operations of WinLC RTX differ from the Windows NT version of WinLC:

- Initial values of non-retentive memory areas after shutting down and restarting WinLC:
  - WinLC running on Windows NT: When you shut down the controller, WinLC saves the values of the non-retentive memory areas. These values are restored when you restart the controller.
  - WinLC RTX: When you shut down the controller, WinLC does **NOT** save the values of the non-retentive memory areas. When you restart the controller, the non-retentive memory areas are set to their initial values.
- Support for external connections through the CP 5613 card:
  - WinLC running on Windows NT: You can configure access points (WinLC_0 to WinLC_8) to use the drivers of the CP 5613 card.
  - WinLC RTX: The Real-time (RTX) drivers for the CP 5613 card that were installed with WinLC RTX do not support the use of access points. To configure external access points for WinLC RTX, you must use a second CP card (such as a CP 5611).
- Support for equidistant DP mode (EDM):
  - WinLC operates in normal mode. In normal mode, the DP cycle and the PLC cycle operate asynchronously to each other.
  - With WinLC RTX 3.1, you can operate the DP Master in normal mode or in constant bus cycle time (equidistant) mode. In constant bus cycle time mode, you can assign a process image partition to the DP master for synchronous update.
- WinLC RTX supports, SFC 82, SFC 83, and SFC 84, which allow you to create, copy and write data to Load memory. WinLC does not have this capability.

Setting Up the WinLC RTX Software 2

Chapter Overview

To use WinLC RTX for process control, you must install and authorize the WinAC RTX software on your computer. In addition, you must have installed a communications processor CP 5613 card in your computer.

Note

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

The Setup program for WinLC RTX configures the CP 5613 card in your computer as the access point for WinLC RTX. 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.

Section	Description	Page
2.1	Overview of the WinLC RTX Installation	2-2
2.2	Installing the WinLC RTX Software	2-3
2.3	Running WinLC RTX as an NT Service	2-5
2.4	Uninstalling the WinLC RTX Software	2-6
2.5	Authorizing the WinAC RTX Software	2-7
2.6	Special Notes for Installing the CP 5613 Card	2-9
2.7	Troubleshooting the Installation of WinLC RTX	2-12

2.1 Overview of the WinLC RTX Installation

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

- CP 5613 card (purchased separately)
- VenturCom Real-time extensions (RTX) for Windows NT
- WinLC RTX software (and other elements of WinAC RTX)

WinLC RTX comes with new real-time drivers for the CP 5613 card.

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

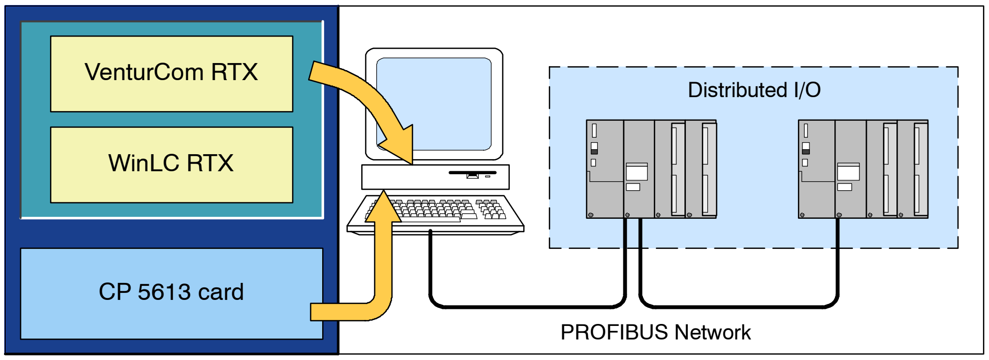


Figure 2-1 Installing the Components for WinLC RTX

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 another SIMATIC software package), refer to the installation procedures for that product.

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

- You must install the CP 5613 card in your computer; however, do not install the SIMATIC NET software. For information about installing the CP card, refer to the documentation for the CP 5613 and to Section 2.6.
- You must use the Setup program for WinAC RTX to install WinLC RTX. The device drivers for the CP 5613 card are included during the installation. See Section 2.2.
- You must authorize WinAC RTX for use on your computer. See Section 2.5.

Note

The Setup program configures the first CP 5613 card in your computer as the access point for WinLC RTX. You do not use the Setting the PG/PC Interface application to configure the CP 5613 card for WinLC RTX.

2.2 Installing the WinLC RTX Software

WinAC RTX 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 RTX. To install only WinLC RTX, deselect the other components of WinAC RTX and select to install only WinLC RTX.

During installation, the program checks to see whether an authorization is installed on the hard disk. If you wish, you can run the authorization program immediately or continue the installation and execute the authorization later. See Section 2.5 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.

Note

Before installing WinLC RTX, the Setup program automatically removes any existing version of WinLC and the DP 5613 drivers from the computer.

Use the following procedure to start the installation program:

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 Install.
3. Insert the CD-ROM and click Next. Windows NT searches automatically for the installation program SETUP.EXE.
4. Follow the instructions displayed by the Setup program and select the elements of WinAC RTX to install:
   - The Setup program first installs the VenturCom Real-time extensions (RTX). After the RTX extensions are installed, the Setup program restarts your computer.
   - After the computer has been restarted, the Setup program installs elements of WinAC RTX that you selected.
5. When prompted by the software, insert the WinLC RTX authorization diskette in drive A. For more information about authorizing the WinLC RTX software, see Section 2.5.

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

Note

You can configure WinLC RTX to connect to STEP 7 that is running either on the same computer as WinLC RTX or on another computer. Refer to Sections 3.1 and 3.2.

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 100 Mbytes of free space on your hard disk for WinLC RTX.
- Bad disk: Verify that the WinAC RTX CD is damaged, then call your local Siemens representative.

2.3 Running WinLC RTX as an NT Service

The Setup program allows you to choose whether to install WinLC RTX as an NT service. You must have administrative privileges to install WinLC RTX as a service. By running as an NT service, WinLC RTX starts automatically any time you start the computer. You can use the Windows NT control panel to change this selection later if you wish. See Section 4.7.

Note

You must have Administrator (ADMIN) privileges to manually start WinLC RTX. To allow WinLC RTX to run when a non-administrator uses the computer, configure WinLC RTX as an NT service. When the non-administrator starts the computer and logs in, the WinLC RTX service runs. The non-administrator can then perform all of the functions allowed by the level of security for WinLC RTX.

WinLC RTX execution can be controlled from the Windows NT Control Panel when WinLC RTX is configured to run as an NT service. To access Windows NT services, follow the procedure below:

1. Select **Start** > **Settings** > **Control Panel** .
2. Double-click on the Services icon to open the Services dialog box.
3. Select “SIMATIC WinLC RTX” from the list of NT Services. Notice that the Startup behavior is listed as Automatic.
4. To start or stop WinLC RTX, use the Start and Stop buttons on the Services dialog box.

To change WinLC RTX so that it does not start automatically after a reboot, follow the procedure below:

1. On the Services dialog box, click the Startup button to display the dialog box.
2. In the Startup Type field, select “Manual” and click OK. Notice that the Startup behavior is now listed as Manual.
3. Close the Services dialog box.

After changing WinLC RTX 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 RTX.

2.4 Uninstalling the WinLC RTX Software

Use the Add/Remove Programs utility of Windows NT to uninstall the WinLC RTX 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. From the list of installed programs, select WinLC RTX and click on the Add/Remove button. Windows NT then uninstalls the WinLC RTX 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 RTX may be irretrievably lost.

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

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

2.5 Authorizing the WinAC RTX Software

WinAC RTX requires a product-specific authorization (or license for use).

Note

If you remove the authorization, the WinLC RTX controller continues to operate.

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

If you install an authorization while the WinLC RTX 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 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 RTX may be irretrievably lost.

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

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

Installing the Authorization

When you install your software for the first time, a message prompts you to install the authorization. Use the following procedure to install the authorization for WinLC RTX:

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 RTX.

If you attempt to start WinLC RTX and there is no authorization available for the software, a message informs you of this. If you want to install the authorization, select **Start** > **Simatic** > **AuthorsW** .> **AuthorsW** . This program allows you to display, install, and remove authorizations. If AuthorsW is not installed on your computer, reinstall WinLC RTX and select the AuthorsW checkbox.

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.

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

1. Insert the original authorization diskette in your floppy disk drive.
2. Select **Start** > **Simatic** > **AuthorsW** .> **AuthorsW** .
3. From the list of all authorizations on drive C, select the authorization to be removed.
4. Select the **Authorization** > **Transfer...** menu command.
5. In the dialog 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.

2.6 Special Notes for Installing the CP 5613 Card

Caution

Attempting to run WinLC RTX with both the WinLC RTX drivers and the SIMATIC NET drivers for the CP 5613 installed on your computer can cause unpredictable operation of the CP 5613 card, which might result in potential damage to equipment and possible injury to personnel.

Do not install the SIMATIC NET software for the CP 5613 card after you have installed WinLC RTX.

The WinLC RTX controller uses a real-time device driver to access the CP 5613 card. This device driver replaces the SIMATIC NET CP 5613 device driver delivered with the CP 5613 hardware.

The SIMATIC NET software for the CP 5613 includes the following products: DP 5613, S7 5613, FMS 5613, and CP 5613/ CP 5614 Software DP Base. Installing any of these products on your computer installs the SIMATIC NET device drivers for the CP 5613 card.

The Setup program for WinLC RTX removes any existing CP 5613 device drivers from your computer before installing the WinLC RTX device drivers for the CP 5613 card. Do not install the CP 5613 software from the SIMATIC NET CD, especially after you have installed the WinLC RTX software.

Accessing the CP 5613 in Polled or Interrupt Mode

WinLC RTX accesses the CP 5613 card in either “Polled” or “Interrupt” mode. Interrupt mode provides improved performance over Polled mode.

Note

When the CP5613 is operating in polled mode, you will not be able to use it for synchronous I/O updates (equidistant DP). System response to hardware interrupts (OB40), diagnostic interrupts (OB82), module plug/pull interrupts (OB83), and station failure/return interrupts (OB86) will be delayed up to 2 ms per interrupt. Also, throughput for module parameterization and data set read/writes to modules is reduced.

To ensure that WinLC RTX accesses the CP 5613 in interrupt mode, you need a CP5613 card (Rev. 3 or greater) installed in a PCI slot that does not share an IRQ number with any Windows-controlled device.

Use the following procedure to determine the IRQ number assigned to the CP 5613:

1. Browse to the Program Files\Vci\RTX\Samples directory (typically on the C drive) and locate the ScanBus.rtss utility.
2. Double-click on the ScanBus.rtss icon. The ScanBus utility lists all of the PCI devices which have been installed on your computer. Included in this list are the resources for each device.
3. Locate the PciData for the following device:
   - VendorID: 0x110a
   - DeviceID: 0x3142
4. The InterruptLine entry shows the IRQ number that has been assigned for the CP 5613 card. Use this number to resolve the IRQ assignment conflict, following the procedure below.

Note

If the requirement for a CP5613 card (Rev. 3 or greater) with a unique IRQ number is not met, a warning message displays each time the CP5613rtx driver is initialized. The warning states that the CP5613 card is operating in polled mode and indicates the configured IRQ number of the CP5613. Use this number to resolve the IRQ assignment conflict, following the procedure below.

To resolve an IRQ assignment conflict with the CP5613, display the IRQ numbers assigned to the NT devices on your computer:

1. Select the **Start** > **Programs** > **Administrative Tools (Common)** > **Windows NT Diagnostics** menu command to display the Windows NT Diagnostics dialog box.
2. Select the Resources tab to display the IRQ numbers assigned to the devices installed in your computer:
   - If there is an entry for “cp5613” in the device list, you have installed a component of the SIMATIC NET CP 5613 software. You must remove this software before WinLC RTX can function correctly.
   - Compare the IRQ numbers listed for the NT devices with the IRQ number of the CP 5613.

If the IRQ for the CP 5613 card is assigned to an NT device, use one of the following methods to change the system configuration for your computer and assign a different IRQ number to the CP 5613 card:

- Use the BIOS setup utility of your computer to assign a unique IRQ to the CP 5613 card.
- Install the CP 5613 card in a different slot on the PCI bus of your computer. You may need to disable an on-board facility in order to free an IRQ for use by the CP5613.

Note

This may require an iterative process. For some computers you may not be able to resolve the IRQ conflict. If you have problems assigning a unique IRQ number to the CP 5613 card, contact your PC vendor for more information.

2.7 Troubleshooting the Installation of WinLC RTX

Problems Occur If You Have Not Installed Windows NT 4.0 Service Pack 6

The VenturCom Real-time extensions (RTX) and WinLC RTX require Windows NT version 4.0 Service Pack 6 (SP6) in order to operate properly. Attempting to run WinLC without this version of Windows NT might cause problems, including the following:

- 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 6 is available as a free download from Microsoft (www.microsoft.com).

If You Cannot Start the WinLC Controller from the WinLC Control Panel

When you install WinLC RTX, the Setup program creates and registers the Active File, which is an important WinLC system file. If for any reason the Active File path is not found in the registry, the WinLC RTX controller cannot be started from the WinLC control panel.

Use the following procedure to restore the path for the Active file:

1. Use Windows Explorer to start the WinLC RTX controller:
   - Locate the executable file for the controller:

     S7wlcrtx.exe
   - Double-click on its icon to start the controller. (This action starts only the WinLC RTX controller; it does not display the WinLC control panel.)

   As WinLC RTX starts the controller, WinLC RTX registers the Active File path for you. WinLC RTX opens an empty controller (no user program) in STOP mode.
2. To connect to the controller, double-click the WinLC RTX icon on your desktop to display the WinLC control panel.

Note

If you use this method to start the WinLC RTX controller, you will not be able to shut down the WinLC RTX controller from the application. You must log out of the Windows NT session in order to shut down the WinLC RTX controller.

Problems Occur When You Install Elements of WinAC While WinLC RTX Is Running

Do not install any elements of WinAC (such as SIMATIC Computing) on a computer that is actively running WinLC RTX during the installation. Since these products use the common resources, this could cause the files to become corrupted. Always stop the execution of WinLC RTX (or any other element of WinAC) before installing any of the WinAC software.

Caution

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

Since SIMATIC Computing, WinLC RTX, 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 RTX:

- WinLC
- Panel for CPU 416-2 DP ISA or any other slot PLC
- SIMATIC Computing SoftContainer
- TagFile Configurator
- Toolmanager
- 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

Problems with the Autostart Feature

In order to use the Autostart feature for modifying the start-up and shut-down behavior of WinLC RTX, you must have administrator privileges. If you do not have administrator privileges, you cannot modify the Autostart options.

See Section 4.7 for more information about the Autostart feature.

Problems Occur If You Use STEP 7 to Change the MPI Address for WinLC RTX

Using the hardware configuration application of STEP 7 to change the MPI node address for WinLC RTX causes communication problems on your MPI network. These problems occur because the logical address (as configured by STEP 7) does not match the physical address for WinLC (as determined by the MPI card, typically MPI=2).

The MPI address of WinLC RTX (MPI=2) is determined by the MPI card installed in your computer. This hardware-configured MPI address for WinLC RTX is independent of any MPI address that you can configure in STEP 7. Always leave the node address for WinLC RTX set to MPI=2.

Problems Occur If You Uninstall (Remove) the Computing Software While WinLC RTX Is Running

If WinLC RTX is being executed during the uninstallation procedure for the Computing software, WinLC RTX experiences a connection error and loses connection to the machine or process. Use the following procedure to recover from the connection error:

1. Using the Windows NT Task Manager, end the process for WinLC RTX (S7wlcrtx.exe).
2. If the WinLC control panel is open, close the control panel.
3. Restart WinLC RTX to reconnect to the machine or process.

Warning

Uninstalling (removing) the Computing software at the same time that WinLC RTX is being executed on that computer causes WinLC RTX 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 RTX to lose connection to the process by uninstalling Computing, use the Windows NT Task Manager to end the WinLC RTX process (S7wlcrtx.exe). If the WinLC control panel is open, close the panel. To reconnect WinLC RTX to the machine or process, restart WinLC RTX.

Before removing the Computing software, always ensure that the WinLC RTX controller has been shut down and that the WinLC RTX software is not being executed. This helps to ensure that you do not cause WinLC RTX 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.

Connecting SIMATIC Client Software to WinLC RTX 3

Chapter Overview

WinLC RTX allows you to connect to SIMATIC products, such as STEP 7, WinCC, and ProTool Pro, across networks using MPI, PROFIBUS-DP, or Ethernet (H1) networks. For more information, refer to the documentation for the specific products.

This chapter provides information about different configurations for connecting WinLC RTX to STEP 7.

3.1 Connecting STEP 7 to WinLC RTX on the Same Computer

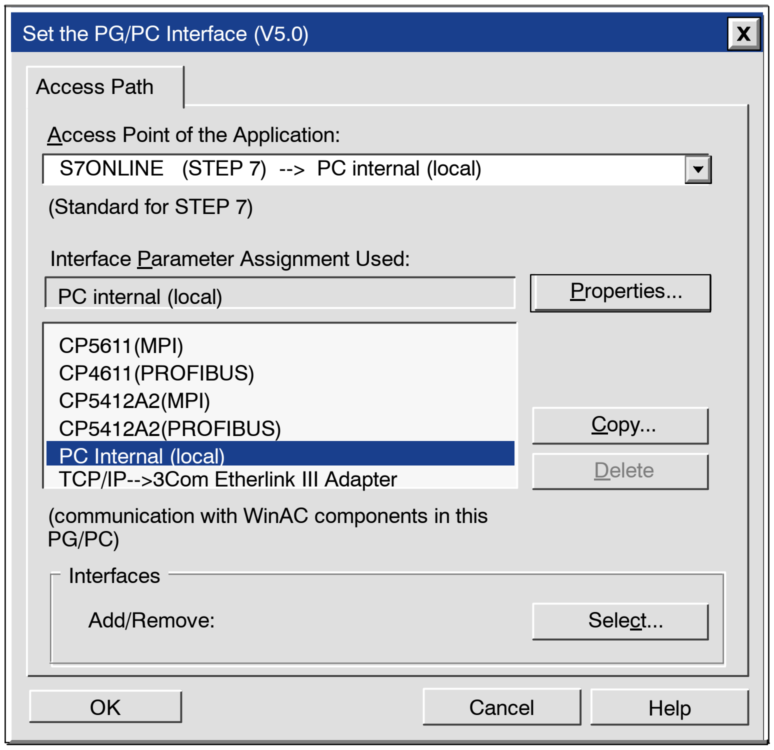
Use the following procedure to configure STEP 7 for communicating with WinLC RTX on the same computer:

1. From WinLC RTX, select the **CPU > Setting the PG/PC Interface** menu command to open the Set the PG/PC Interface dialog box.
2. As shown in Figure 3-1, select the following entry from the drop-down list for the "Access Point of the Application" field:

   S7ONLINE (STEP 7)
3. For the "Interface Parameter Assignment Used" field, select the following parameter:

   PC Internal (local)
4. Click OK to enter the configuration and close the dialog box.

STEP 7 is now configured to communicate with WinLC RTX on the local 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)

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

3.2 Connecting STEP 7 to WinLC RTX on a Different Computer

As shown in Figure 3-2, you can connect STEP 7 on one computer to WinLC RTX on a different computer.

- You must define the network connection over which STEP 7 and WinLC RTX 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 RTX installed.

Note

To configure STEP 7 and WinLC RTX for H1 communications, you must also have installed the following software:

- NCM Options package for H1 communication
- STEP 7 version 5 service pack 3 (SP3)

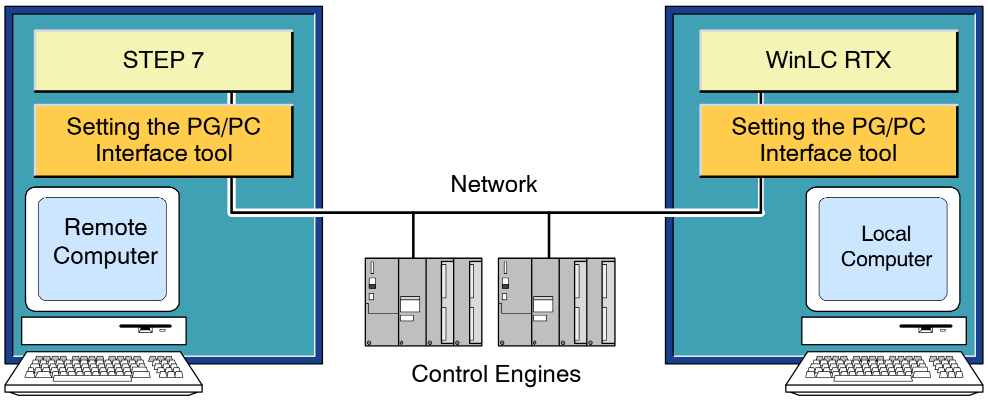


Figure 3-2 Connecting STEP 7 to WinLC RTX over a network

Note

NetPro cannot reconfigure the MPI or H1 addresses or the bus parameters of a WinLC RTX from a different computer. The required CP cards are not controlled by WinLC RTX. This can be done only by means of the local Setting the PG/PC Interface application. The PROFIBUS node address and bus parameters can be reconfigured remotely. WinLC RTX is the master for the CP 5613 card.

Configuring the Computer with STEP 7

You must configure the computer on which STEP 7 resides for communicating with WinLC RTX on a remote computer:

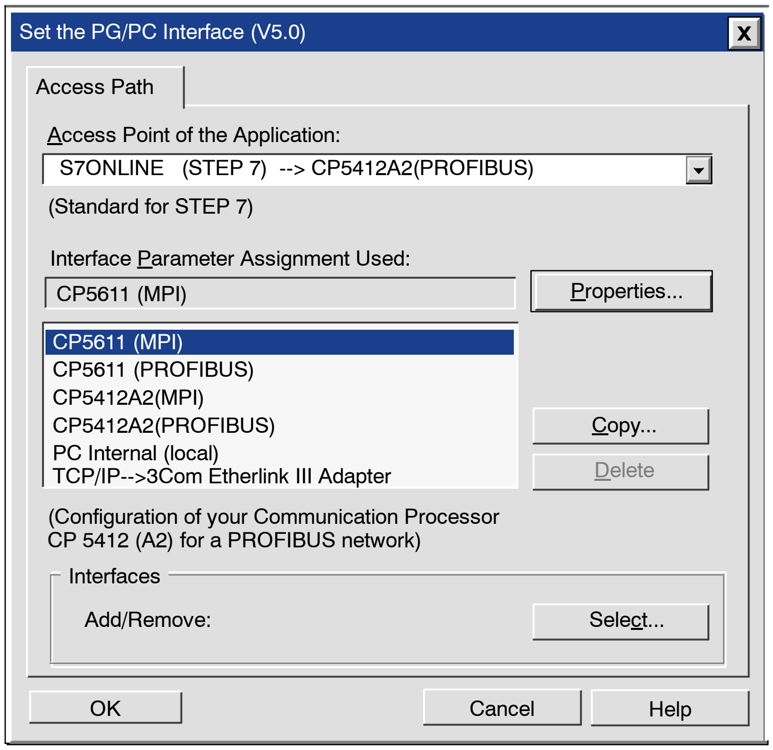
1. From SIMATIC Manager, select the **Options > Set the PG/PC Interface** menu command to display the Setting the PG/PC dialog box.
2. As shown in Figure 3-3, select the following entry from the drop-down list for the "Access Point of the Application" field:

   S7ONLINE (STEP 7)
3. If you are using MPI as your network communication path, select the MPI interface parameter. For example:

   CP5611 (MPI)
4. If you are using PROFIBUS-DP as your network communication path, select the PROFIBUS-DP interface parameter. For example:

   CP5412A2(PROFIBUS)
5. If you are using H1 as your network communication path, select the TCP/IP interface parameter. For example:

   TCP/IP -> 3Com Etherlink III Ada
6. Click OK to save the configuration.



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

Interface Parameter Assignment Used:
CP5611 (MPI)

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 the Computer with STEP 7 Installed

Note

Before WinLC RTX is visible to other programming devices on the PROFIBUS-DP network, you must use the Setting the PG/PC Interface dialog box to configure the CP card:

S7ONLINE (STEP7) --> PROFIBUS

In addition, you must select the "PG is the only master on the bus" option.

Configuring the Computer where WinLC RTX Is Installed

You must also configure the communication path(s) from the computer on which WinLC RXT resides to networks with computer(s) running STEP 7. WinLC RTX installs nine access points. Each access point can point to one of the installed interfaces. In the following example, WinLC RTX provides access by means of two different CP cards (CP 5412 and CP 5611) at the same time. You cannot access WinLC through any CP cards that are not assigned to an access point.

WinLC_0 --> none

WinLC_1 --> CP5412A2(MPI)

WinLC_2 --> none

WinLC_3 --> none

WinLC_4 --> none

WinLC_5 --> none

WinLC_6 --> CP5611 (MPI)

WinLC_7 --> none

WinLC_8 --> none

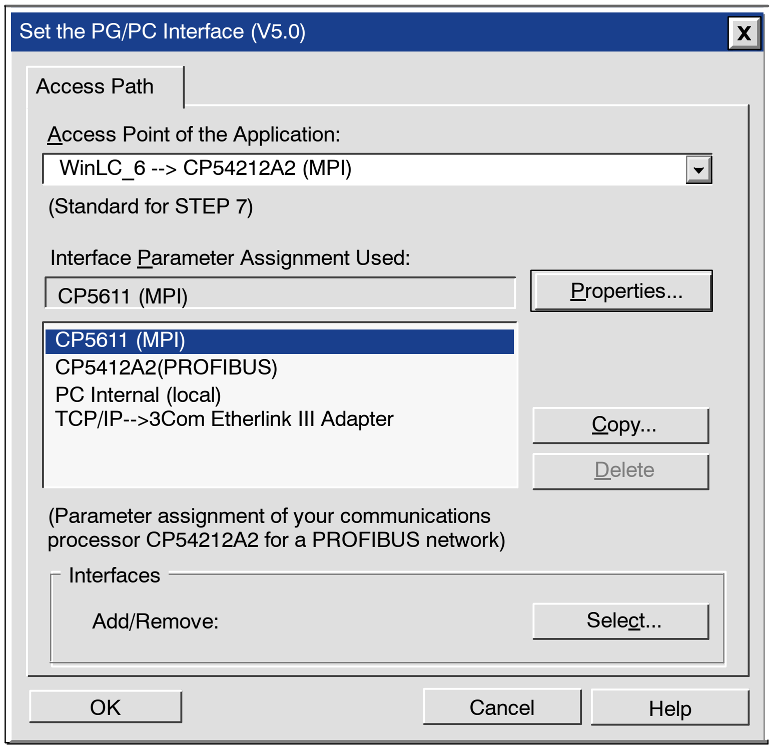
Use the following procedure to configure one of the access points in the example:

1. Using the WinLC RTX control panel, select the **CPU > Setting the PG/PC Interface** menu command to display the Setting the PG/PC dialog box.
2. From the drop-down list of the "Access point of application" field, select the access point. For example: WinLC_6
3. As shown in Figure 3-4, select the interface parameter from the parameter set that corresponds to your network communications path. For example:

   CP5611 (MPI)

Repeat steps 2. and 3. as needed to configure any other access point used to communicate to your network.

4. Shut down and restart the WinLC RTX application so that your selections can take effect. Refer to page 4-19 for the appropriate shutdown and restart procedure.


Set the PG/PC Interface (V5.0)

Access Path

Access Point of the Application:

WinLC_6 --> CP54212A2 (MPI)

(Standard for STEP 7)

Interface Parameter Assignment Used:

CP5611 (MPI)

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

Connecting STEP 7 to Hardware PLCs

After you have configured STEP 7 for communicating with WinLC RTX on the remote computer (Section 3.2) in addition to any hardware PLCs on the network, you can use any of the STEP 7 tools or functionality across the network.

Note

WinLC RTX cannot perform the cyclic distribution of PROFIBUS bus parameters.

Running the WinLC RTX Software 4

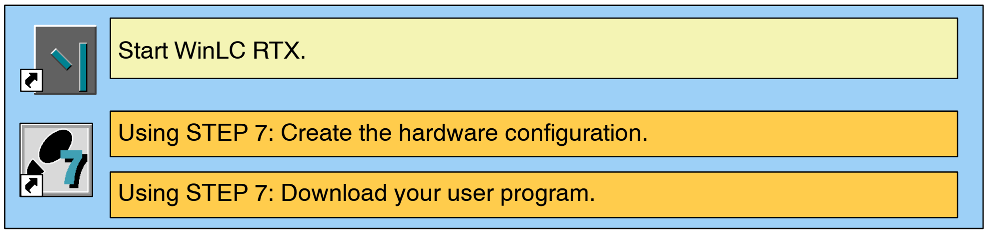
Chapter Overview

The WinLC control panel allows you to control the operation of the WinLC RTX controller by performing the following tasks:

- Monitoring the status
- Changing the operating mode
- Registering and unregistering WinLC RTX as an NT service
- Performing a cold restart or a warm restart
- Enabling the Autostart feature of WinLC RTX
- Monitoring the scan cycle
- Tuning the operations of WinLC RTX
- Changing the language for the WinAC applications
- Creating levels of access and security for WinLC RTX
- Changing the password for WinLC RTX

4.1 Starting the WinLC RTX Software

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



Start WinLC RTX.

Using STEP 7: Create the hardware configuration.

Using STEP 7: Download your user program.

Figure 4-1 Starting WinLC RTX

Note

You must have administrator (ADMIN) privileges to start WinLC RTX. To allow non-administrators to start WinLC RTX, configure WinLC RTX to run as an NT service on the computer. See Section 4.7 and Section 2.3.

Getting Started

Use the following procedure to start WinLC RTX:

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

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

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

Setting Network Connections for STEP 7

From the WinLC control panel, select **CPU > Setting the PG/PC Interface** menu. Chapter 3 describes how to configure network settings.

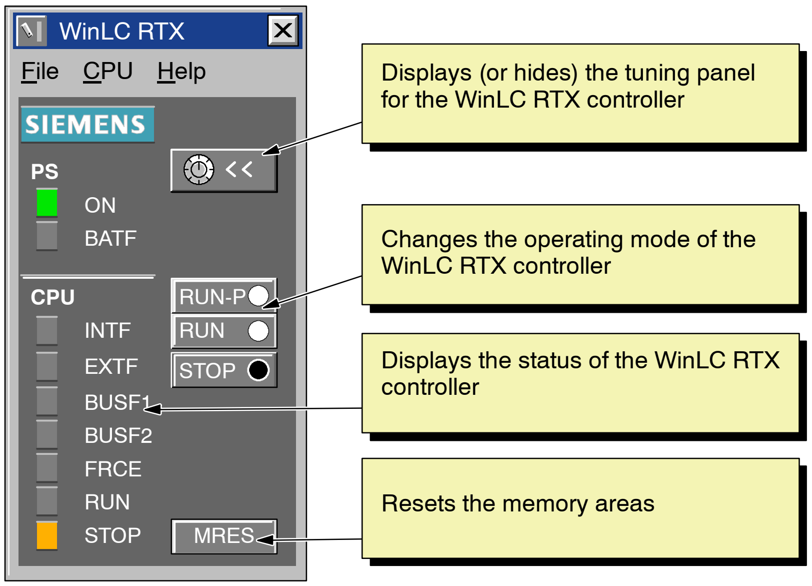

WinLC RTX
File CPU Help
SIEMENS
PS
ON
BATF
CPU
INTF
EXTF
BUSF1
BUSF2
FRCE
RUN
STOP
RUN-P
RUN
STOP
MRES
Displays (or hides) the tuning panel for the WinLC RTX controller
Changes the operating mode of the WinLC RTX controller
Displays the status of the WinLC RTX controller
Resets the memory areas

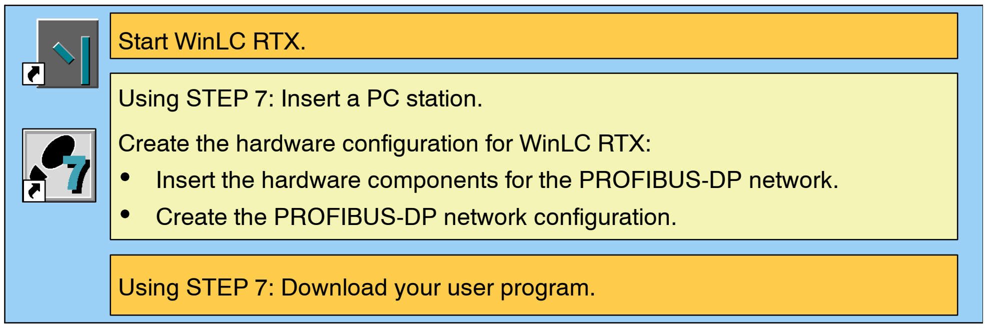
Figure 4-2 Control Panel for WinLC RTX

4.2 Creating the Hardware Configuration

The hardware configuration defines the network addresses and the distributed I/O (DP) for the WinLC RTX 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 RTX:

- Use the SIMATIC Manager to create a project and a PC station.
- Use the Hardware Configuration to configure WinLC RTX 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 RTX.

Using STEP 7: Insert a PC station.

Create the hardware configuration for WinLC RTX:

- 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 WinLC RTX

Inserting a Station for WinLC RTX in STEP 7

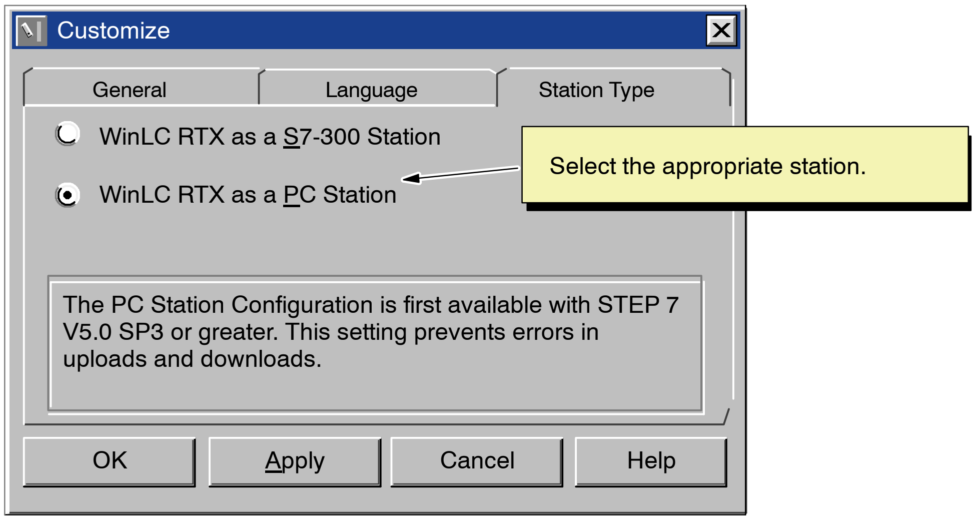
Before you can create the hardware configuration for WinLC RTX, you must insert a station under your project. For STEP 7 Version 5, Service Pack 3, insert a PC station. STEP 7 V5 SP3 and later model WinLC RTX as a component in a PC station. (For versions of STEP 7 prior to Version 5, Service Pack 3,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

1. Select (click on) the project. For example, select the sample 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 the **Insert > SIMATIC 300 Station** menu command.)
3. Select the station to display the hardware icon for the station.

Note

Certain System Data Blocks have a different structure, depending on whether WinLC RTX is configured in a 300 Station or a PC Station. You must manually select the correct station type for your application. Failure to configure the correct station type will cause upload/download error messages to be displayed.

1. From the WinLC control panel, select the **CPU > Options > Customize** menu command.
2. Select the Station Type tab on the Customize dialog box**.**
3. As shown in Figure 4-4, select the appropriate station type.
4. Click OK to enter the configuration.



( ) WinLC RTX as a S7-300 Station
(●) WinLC RTX as a PC Station

The PC Station Configuration is first available with STEP 7 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 RTX:

Note

Because WinLC RTX has no effect on any installed MPI card, you cannot use the MPI node of WinLC RTX to configure hardware. Do not change the MPI address of WinLC RTX from 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. (See Figure 4-5.)

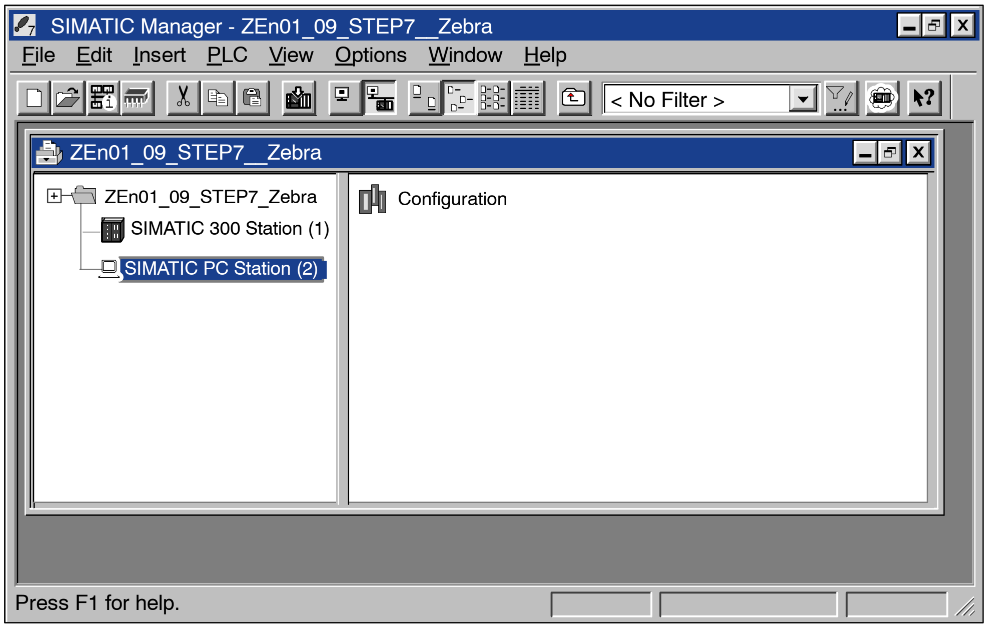


Figure 4-5 Configuring the PC Station in the Sample Project

3. For a PC station:
   - Double-click on the Configuration icon to open the hardware catalog.
   - Select the second slot in the PC configuration table.
   - Select the **SIMATIC PC Station** > **Controller** entry from the catalog. Click on the WinLC RTX icon.
   - Use the mouse to drag the WinLC RTX object into slot 2 of the PC display.

Note

With STEP 7 V5.0 SP3 or higher, WinLC RTX is configured as a PC station. WinLC RTX 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 S7-300 station and configure WinLC RTX 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 New 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 OK to enter the default parameters for a PROFIBUS subnet.
7. Select the PROFIBUS(1) subnet.
8. Click OK to enter the default subnet and address and to close the Properties - PROFIBUS Node DP Master dialog box. WinLC RTX 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 RTX.

STEP 7 generates the system data (SDBs) for the hardware configuration. Exit the Hardware Configuration tool.

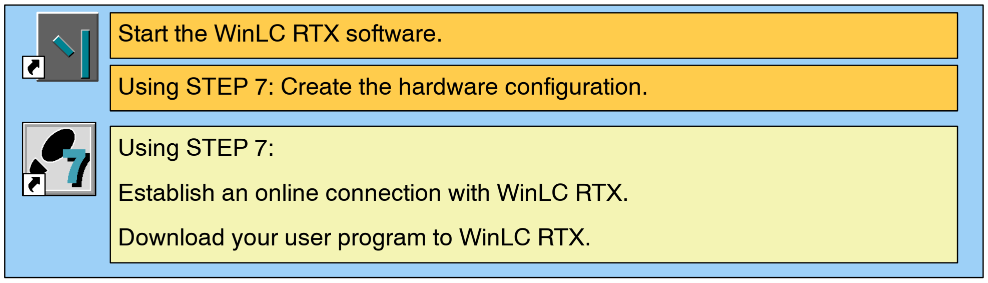
4.3 Downloading Your User Program

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

Due to limitations of the Microsoft “structured document,” programs downloaded to WinLC RTX are limited to 2500 blocks.

Note

If you download 2500 blocks to WinLC RTX, you cannot then replace one of the blocks by downloading a new version unless you first delete the block in the WinLC RTX controller. If you have a user program that consists of 2500 blocks, always delete a block first before downloading another version of that block.



Start the WinLC RTX software.

Using STEP 7: Create the hardware configuration.

Using STEP 7:

Establish an online connection with WinLC RTX.

Download your user program to WinLC RTX.

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

Accessing WinLC RTX from STEP 7

To access WinLC RTX 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 RTX.

Downloading a User Program from STEP 7

After you have established an online connection to WinLC RTX, 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 (SDBs), to WinLC RTX. 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.

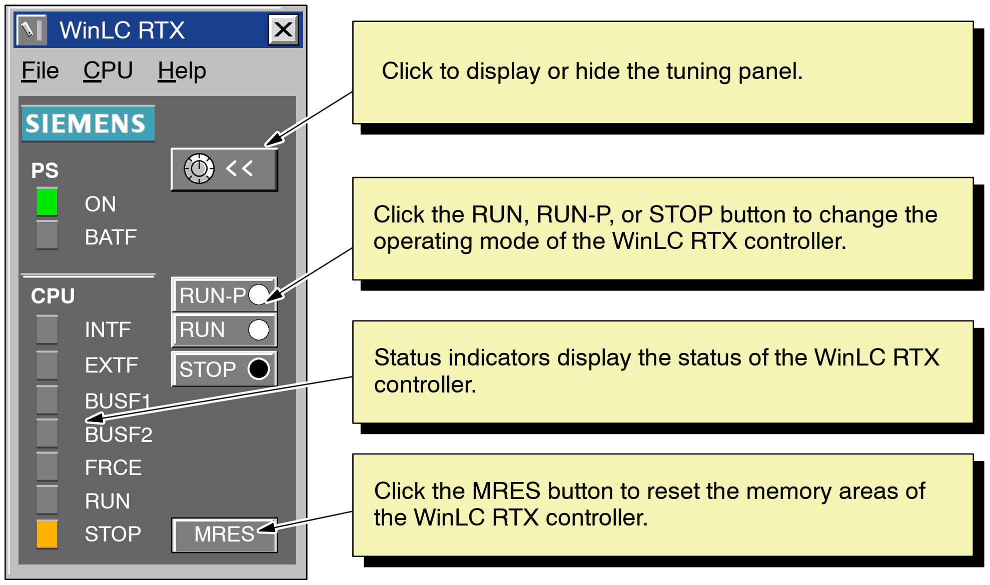
4.4 Executing the User Program

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

Using the Control Panel

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

- A button for displaying or hiding the Tuning Panel that adjusts the operation of WinLC RTX (see Section 4.6)
- Three buttons for changing the operating mode of the controller
- Status indicators for the WinLC RTX controller
- A button for resetting the memory areas



Click to display or hide the tuning panel.

Click the RUN, RUN-P, or STOP button to change the operating mode of the WinLC RTX controller.

Status indicators display the status of the WinLC RTX controller.

Click the MRES button to reset the memory areas of the WinLC RTX controller.

Figure 4-7 Using the Control Panel of WinLC RTX

Selecting the Operating Mode

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

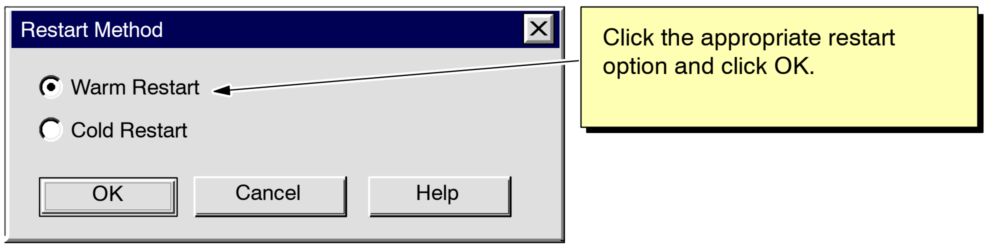
- In STOP mode, the controller is not executing the user program. To download a program that includes SDBs, you must place WinLC RTX in STOP mode. On the transition to STOP mode, the outputs go to a safe state (as configured with the STEP 7 programming software).
- In RUN mode, the controller executes 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.

Click the button to place the controller in the selected operating mode. The status indicators on the control panel show whether the controller is in RUN mode or in STOP mode.

Selecting a Warm Restart or a Cold Restart

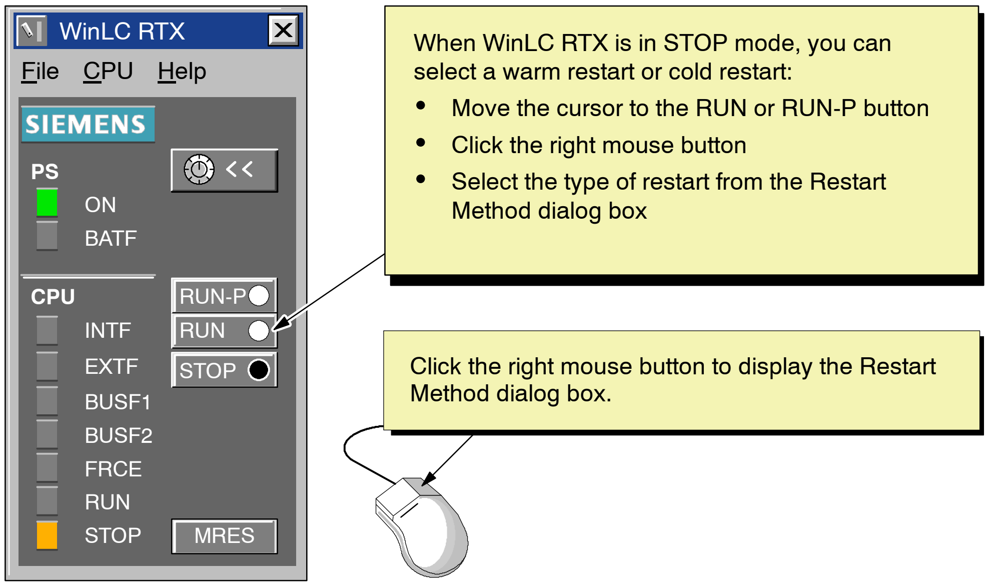
The hardware configuration downloaded with your user program determines the default startup mode for WinLC RTX. (For more information about the restart options, see Section 5.5.) When changing the operating mode of WinLC RTX from STOP mode to RUN mode, you can selectively change the type of restart:

- When you use the menu commands (**CPU** > **RUN** or **CPU** > **RUN-P**) to change the operating mode, WinLC RTX 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 OK.
- Click on the RUN or RUN-P buttons on the control panel to perform a warm restart without displaying the Restart Method dialog box.
- As shown in Figure 4-9, click on the RUN or RUN-P buttons to display the Restart Method dialog box that allows you to select a cold restart or a warm restart.



Click the appropriate restart option and click OK.

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



When WinLC RTX 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 control panel provides a MRES button for resetting the memory areas to the default values and deleting the user program. Click on the MRES button to place the controller in STOP mode and perform 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 RTX Scan Cycle

As shown in Figure 4-10, the WinLC RTX scan cycle begins and ends when WinLC begins writing the process output image to the peripheral output modules. The "free cycle" is the part of the scan cycle that includes OB1 and process image updates. The free cycle consists of consists of four basic processes:

- WinLC RTX writes the status of the process-image output table (the Q memory area) to the outputs. For the first scan, WinLC RTX does not write to the outputs. After the first scan, all other scans start by writing the process-image output table to the outputs.
- WinLC RTX reads the states of the I/O module inputs into the process-image input table (the I memory area).
- WinLC RTX executes the user program in OB1.
- WinLC RTX waits until the free cycle time has elapsed and triggers the next free cycle. This time between completing OB1 execution and starting the next free cycle is the "sleep time" or free cycle wait time.

Note

Although the next free cycle does not start until the end of the sleep (wait) interval, interrupt OBs with a higher priority than the free cycle can execute during the sleep interval if the start event for the OB occurs.

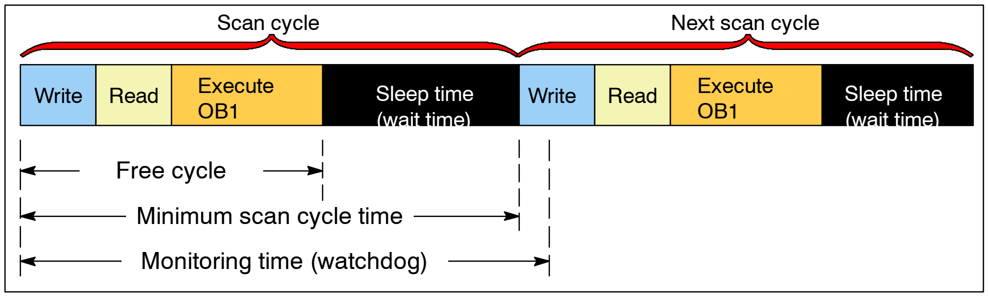


Figure 4-10 Elements of the WinLC RTX Scan Cycle

Configuring the Elements of the Scan Cycle

Use the Min Cycle Time parameter if you want the free cycle to execute on a fixed schedule. Otherwise, leave this parameter at the default value of zero and modify the Min Sleep Time as needed. This ensures that there will always be a wait interval even if the free cycle execution time changes, especially during control program development. You can tune and permanently save the Min Sleep Time parameter from the Tuning Panel. You do not need to use the Hardware Configuration Editor to make the change permanent.

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 RTX.

Caution

Do not set the minimum scan cycle time to be longer than the scan cycle monitoring time (the watchdog time) configured in the STEP 7 Hardware Configuration Editor. Setting the minimum scan time to a value the same as or larger than the watchdog time results in a scan cycle time overrun (WinLC RTX goes to STOP mode during the first scan at the end of the watchdog time interval).

Note

WinLC RTX 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 free cycle time:

- WinLC RTX executes other OBs (such as OB20 and OB35) with higher priorities than OB1.
- Another RTX application that is running on your computer has a higher priority.
- You use a variable table (VAT) with the STEP 7 programming software to display the status of the user program.
- 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.

For more information about modifying sleep time or minimum scan cycle time, see Section 4.6 and Appendix D.

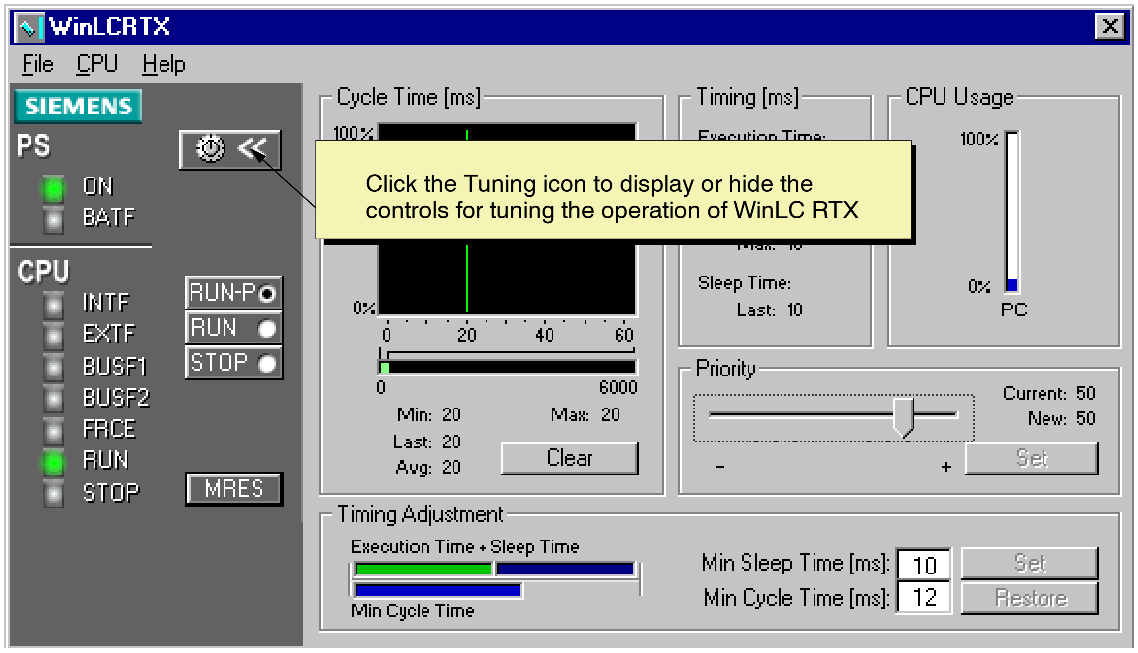
4.6 Tuning the Operation of WinLC RTX

As a member of the SIMATIC S7-300/S7-400 family of programmable logic controllers, WinLC RTX executes control programs created with STEP 7 using the same program execution model as the hardware PLCs. This means that while it is in RUN mode, WinLC RTX executes the control program in continuous repeated scan cycles. Unlike hardware PLCs that are totally dedicated to the PLC functions, WinLC RTX is a software PLC and must share PC CPU time to allow other applications to execute.

WinLC includes a Tuning Panel that you use to configure and test the unique WinLC features for sharing PC CPU time with other applications.

Displaying the Tuning Panel

To display the Tuning Panel, click on the Tuning icon on the control panel or select **CPU > Tuning Panel** (Figure 4-11). If you have enabled password control, you must enter the password in the dialog box before the tuning panel appears. When you finish tuning WinLC RTX, 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 RTX

Cycle Time [ms]: Min: 20, Max: 20, Last: 20, Avg: 20

Timing [ms]: Sleep Time: Last: 10

Priority: Current: 50, New: 50

Min Sleep Time [ms]: 10
Min Cycle Time [ms]: 12

Figure 4-11 Displaying or Hiding the WinLC RTX Tuning Panel

Using the Tuning Controls to Modify WinLC RTX Operations

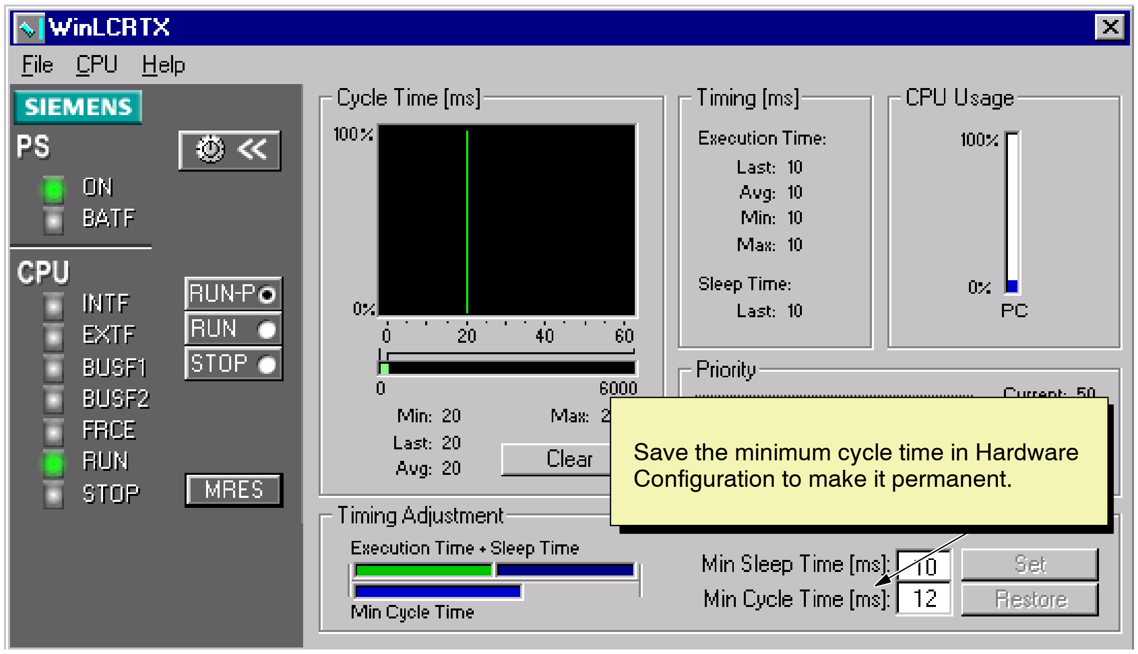
The Tuning Panel (shown in Figure 4-12) allows you to modify elements of the WinLC RTX scan and to set the priority level for the RTX operating system to execute the WinLC RTX software:

- Priority: The slider sets the level of priority for the execution of the program downloaded into WinLC RTX by STEP 7 or from a program archive file. Setting the priority higher for WinLC RTX means that the operating system responds to WinLC RTX before executing lower-priority tasks.

Note

All RTX processes (including WinLC RTX) run at a higher priority than all Windows applications. You only need to adjust the real-time priority for WinLC RTX if additional RTX applications are running with WinLC RTX.

- Timing Adjustment: These fields allow 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 RTX. (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.



Min: 20
Last: 20
Avg: 20
Max: 2

Execution Time:
Last: 10
Avg: 10
Min: 10
Max: 10

Sleep Time:
Last: 10

Min Sleep Time [ms]: 10
Min Cycle Time [ms]: 12

Save the minimum cycle time in Hardware Configuration to make it permanent.

Figure 4-12 Tuning Panel for Adjusting the Operation of WinLC RTX

The tuning panel also provides the following status information:

- Cycle Time (ms) provides a histogram of execution times (in a 120ms range) of the WinLC scan times. 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. A STOP-to-RUN transition also resets the Cycle Time display, as does closing and reopening the tuning pane
- Timing (ms) is a read-only field that displays a summary of the two components of the scan cycle: execution time (actual execution time of all OBs plus I/O image update) and sleep time. Execution time includes: execution time for the last (most current) scan, the average scan cycle time, the shortest (Minimum) scan cycle time, and the longest (Maximum) scan cycle time. Sleep time is the amount of sleep time for the last (most current) scan.
- CPU Usage displays the percentage of the CPU used by the Windows operating system. Because RTX applications (including WinLC RTX) run separately from NT, the CPU usage figure does not reflect RTX usage.

Note

When you use the tuning panel to enter a new value for the Minimum cycle time parameter, WinLC RTX does not change the configured value stored in the control program. WinLC RTX resets the minimum cycle time parameter to the configured value each time you change WinLC RTX from STOP to RUN mode. After you have determined the optimum value of the minimum cycle time, use the STEP 7 Hardware Configuration Editor to change the value saved in the control program. See Sections 4.2 and 5.5.

4.7 Running the WinLC RTX Controller

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

If you do not run WinLC RTX as an NT service, the control panel allows you to start and stop WinLC RTX. An Autostart feature allows you to start WinLC RTX 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 RTX as a service. When you run WinLC RTX as an NT service, you start or stop WinLC RTX either from the Services dialog box or by turning your computer on or off. The control panel does not start or stop WinLC RTX.

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 RTX as an NT Service

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

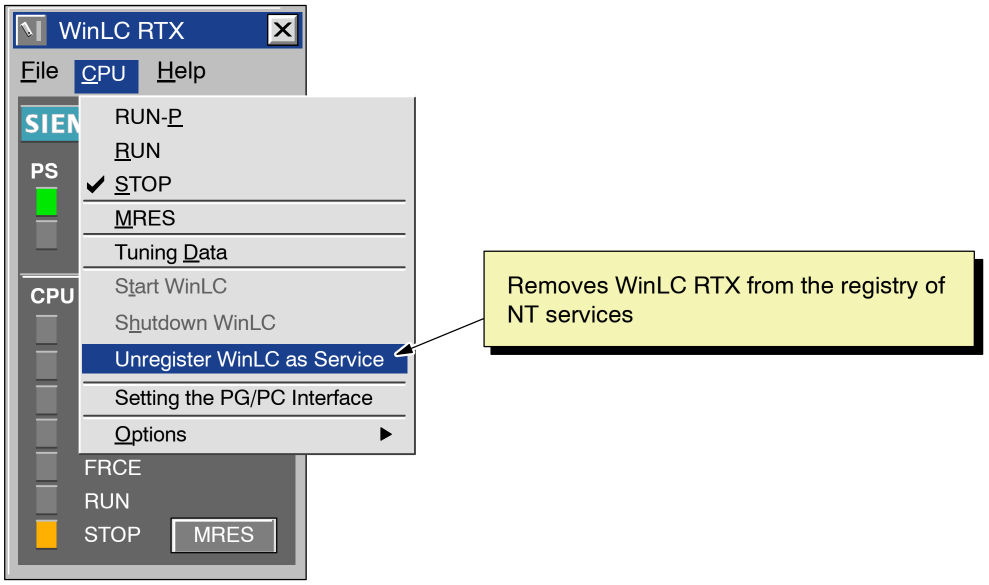

WinLC RTX
File
CPU
Help
SIEN
PS
CPU
RUN-P
RUN
STOP
MRES
Tuning Data
Start WinLC
Shutdown WinLC
Unregister WinLC as Service
Setting the PG/PC Interface
Options
FRCE
RUN
STOP
MRES
Removes WinLC RTX from the registry of NT services

Figure 4-13 Unregistering WinLC RTX as an NT Service

Shutting Down and Starting the WinLC RTX Controller

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

Note

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

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

To start the WinLC RTX controller, select the **CPU** > **Start WinLC** menu command from the WinLC control panel.

If WinLC RTX is running as an NT service, you must use the NT Services dialog box (**Start** > **Settings** > **Control Panel**) to start and stop the service “Siemens WinLC RTX.”

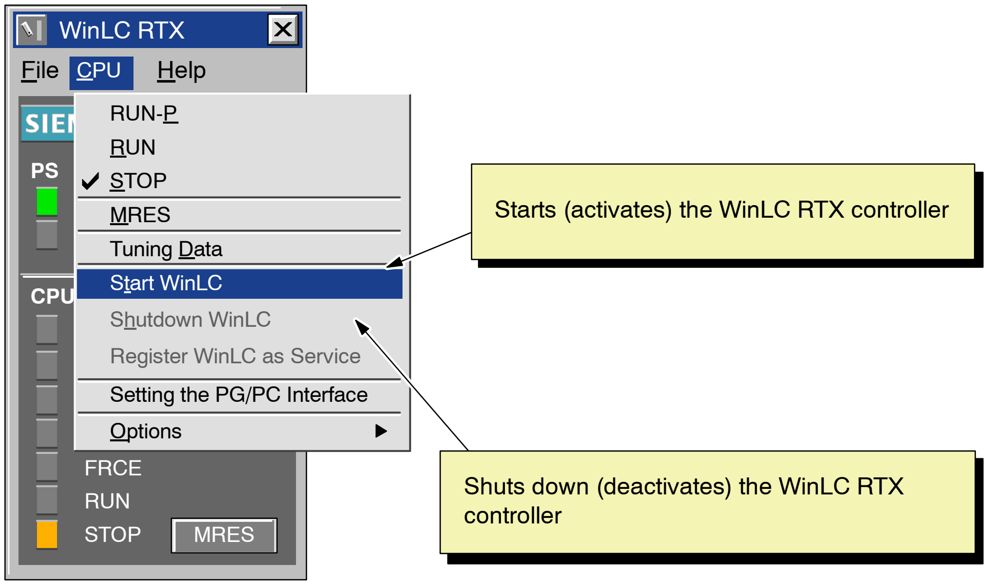

WinLC RTX
File
CPU
Help
SIE
PS
CPU
RUN-P
RUN
✓ STOP
MRES
Tuning Data
Start WinLC
Shutdown WinLC
Register WinLC as Service
Setting the PG/PC Interface
Options
FRCE
RUN
STOP
MRES
Starts (activates) the WinLC RTX controller
Shuts down (deactivates) the WinLC RTX controller

Figure 4-14 Shutting Down the WinLC RTX Controller

Selecting the Autostart Feature

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

If the WinLC RTX controller was running at shutdown ...	And the Autostart feature is selected ...	Then WinLC RTX 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 RTX:

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

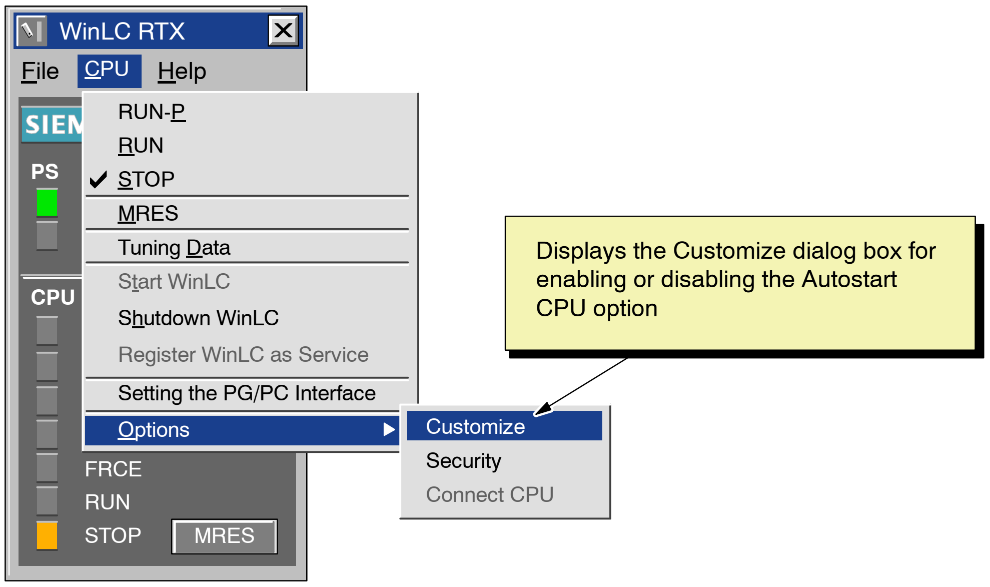
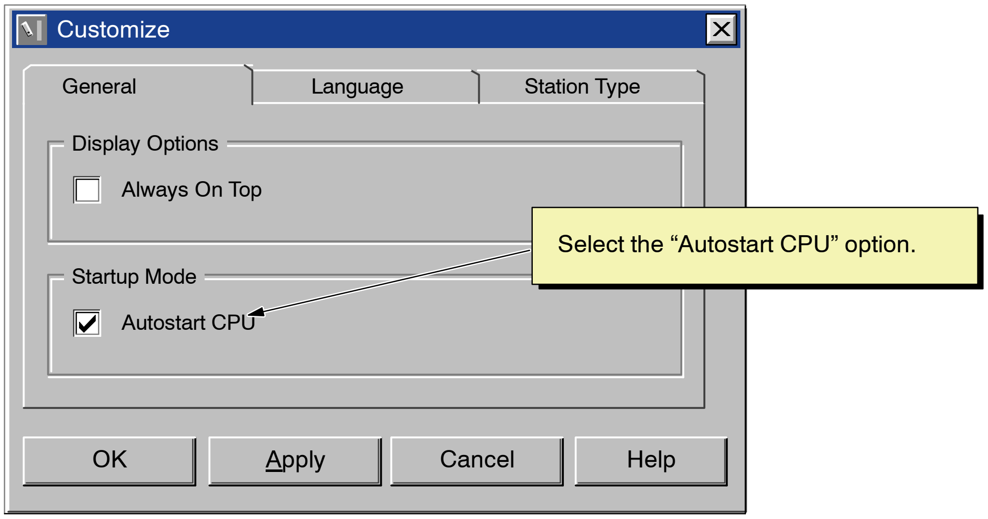

WinLC RTX
File
CPU
Help
SIEM
PS
CPU
RUN-P
RUN
✓ STOP
MRES
Tuning Data
Start WinLC
Shutdown WinLC
Register WinLC as Service
Setting the PG/PC Interface
Options
Customize
Security
Connect CPU
FRCE
RUN
STOP
MRES
Displays the Customize dialog box for enabling or disabling the Autostart CPU option

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 Apply to enable the Autostart feature.
4. Click OK 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 RTX

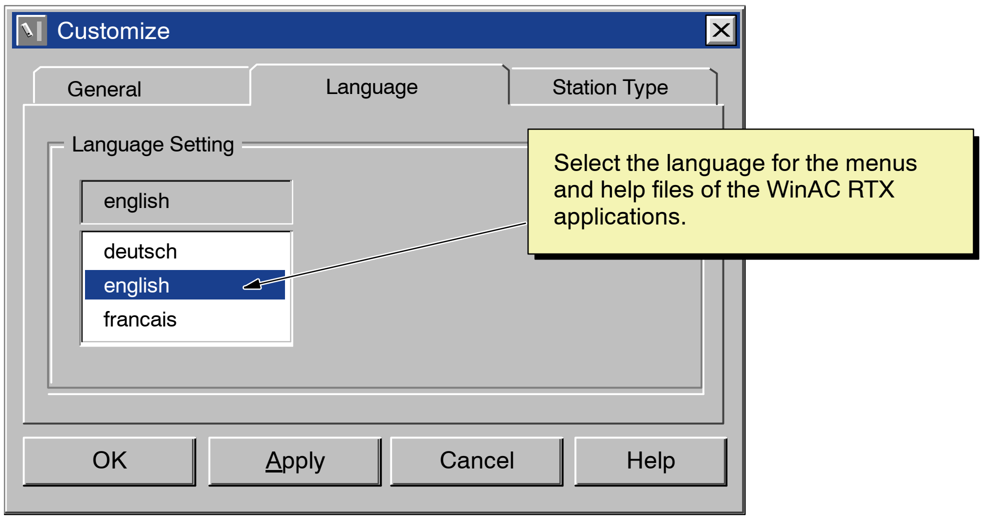
WinAC RTX provides three languages for the software and help: German, English, and French. The menus and help for WinLC RTX are displayed in the language selected. You can change the language from the control panel of WinLC RTX.

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 WinAC RTX applications (German, English, or French). See Figure 4-17.
4. Click Apply to change the language.
5. Click OK to close the Customize dialog box.

Note

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



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

Figure 4-17 Selecting the Language for the WinLC RTX Control Panel and Help Files

4.9 Creating Levels of Security for Access to WinLC RTX

You use the WinLC RTX control panel to create levels of security and limit access to WinLC RTX:

- Select the security level: You can set WinLC RTX 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 WinLC RTX control panel. See Figure 4-18.

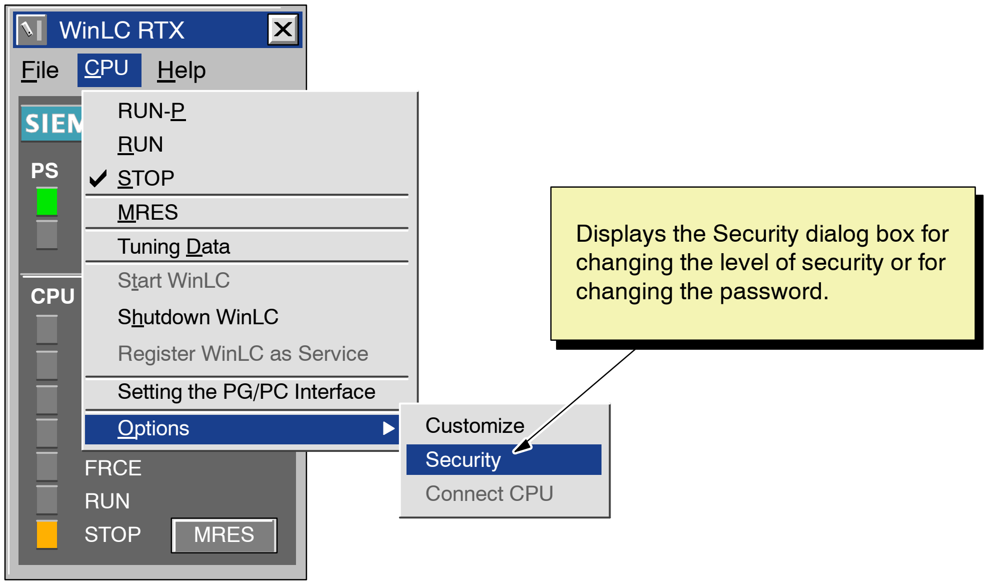

WinLC RTX
File
CPU
Help
SIEM
PS
CPU
RUN-P
RUN
STOP
MRES
Tuning Data
Start WinLC
Shutdown WinLC
Register WinLC as Service
Setting the PG/PC Interface
Options
Customize
Security
Connect CPU
FRCE
RUN
STOP
MRES
Displays the Security dialog box for changing the level of security or for changing the password.

Figure 4-18 Accessing the “Security” Dialog Box

Warning

Running the WinLC RTX 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 death or serious 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 RTX. Always install a physical emergency stop circuit for your machine or process.

Changing the Security Level for WinLC RTX

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

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

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

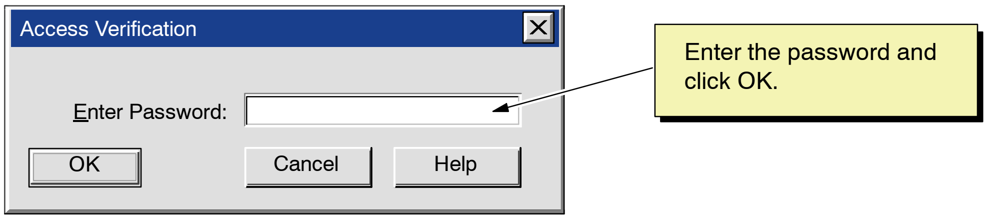


Figure 4-19 Entering the Password for WinLC RTX

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

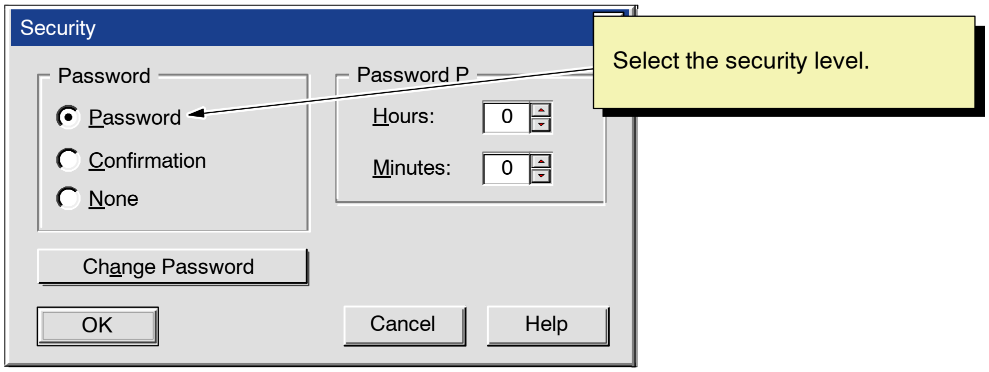


Figure 4-20 Setting the Security Level for WinLC RTX

Creating or Changing the Password for WinLC RTX

The Security dialog box allows you to create or to change the password for WinLC RTX. 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 RTX and click OK. (If the security level is set to “None” or no password has been configured, simply click OK.)
3. In the Security dialog box, click the Change Password button. See Figure 4-21.

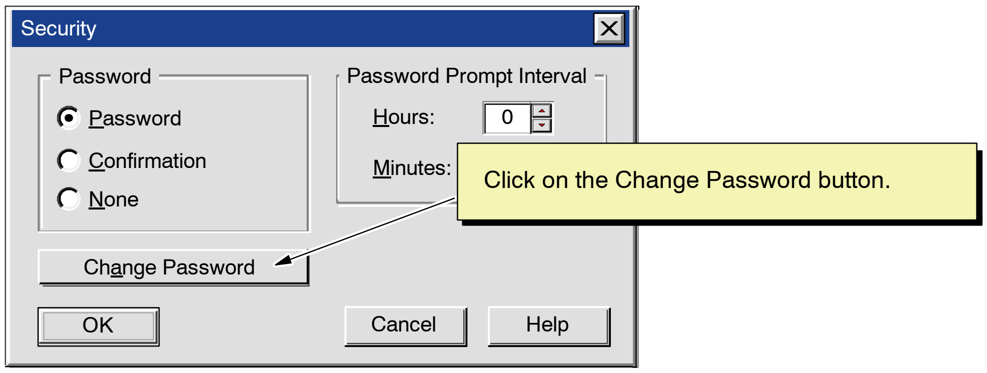


Figure 4-21 Accessing the “Change Password” Dialog Box

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.

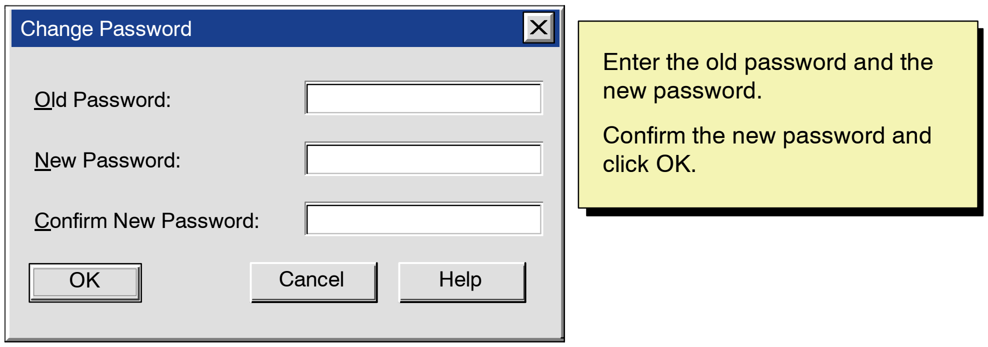

Change Password
Old Password:
New Password:
Confirm New Password:
OK
Cancel
Help
Enter the old password and the new password.
Confirm the new password and click OK.

Figure 4-22 Changing a Password for WinLC RTX

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

Note

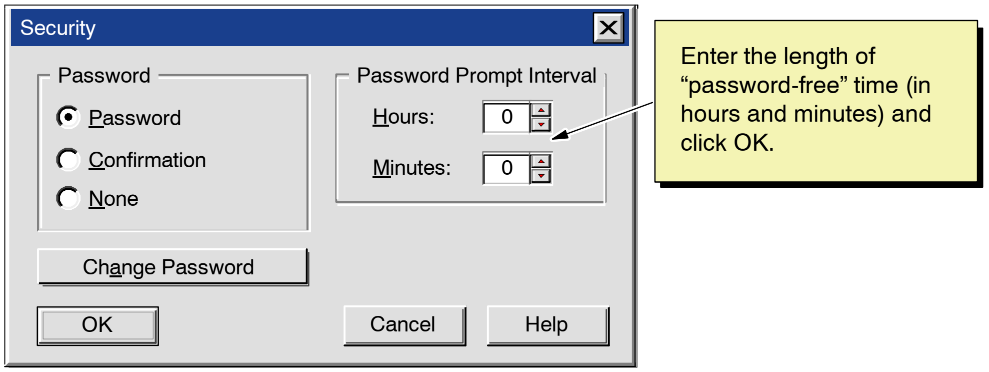
If you create a password, but set the security level to “None” (disabling the password), you will still need to enter the configured password before you can access the Security dialog box again.

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 RTX and click OK. (If the security level is set to “None” or no password has been configured, simply click OK.)
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 OK to enter the length of validity for the password.
5. Make certain that the security level of WinLC RTX is set to the Password option, and click OK to accept the changes and close the Security dialog box.



Password: [✓ Password] Confirmation None

Password Prompt Interval

Hours: 0

Minutes: 0

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

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 control panel. The archive file contains the user program and the hardware configuration (SDBs). A dialog box allows you to save the archive file under a specific name. This allows you to store different archive files.

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, follow these steps:

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

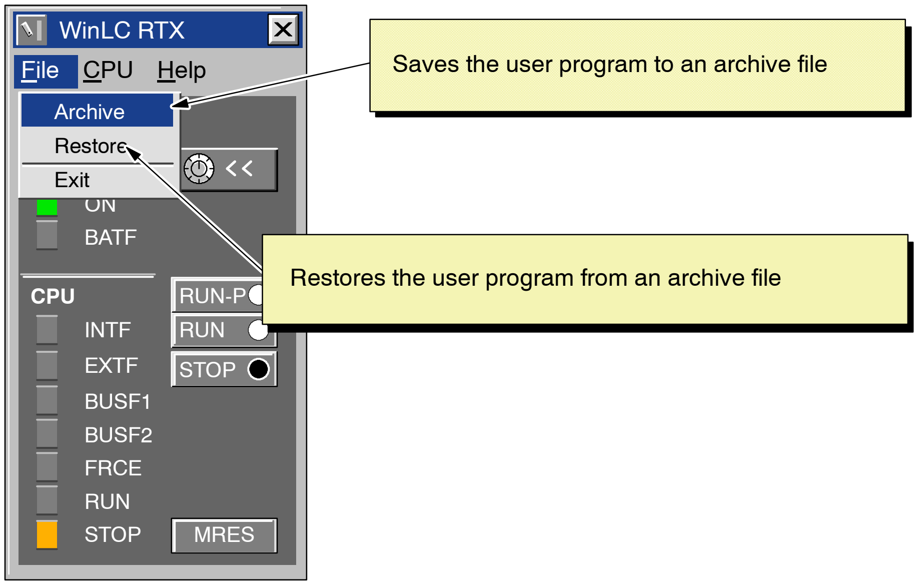


Figure 4-24 WinLC RTX Archive and Restore Commands

Operations of WinLC RTX 5

Chapter Overview

WinLC RTX 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 RTX and includes the following information:

- Elements of the WinLC RTX interface. For additional information, see Chapter 4 and the online help of the WinLC RTX software.
- Memory reset function (MRES) of the PLC 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 RTX 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 WinLC RTX Control Panel

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

Note

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

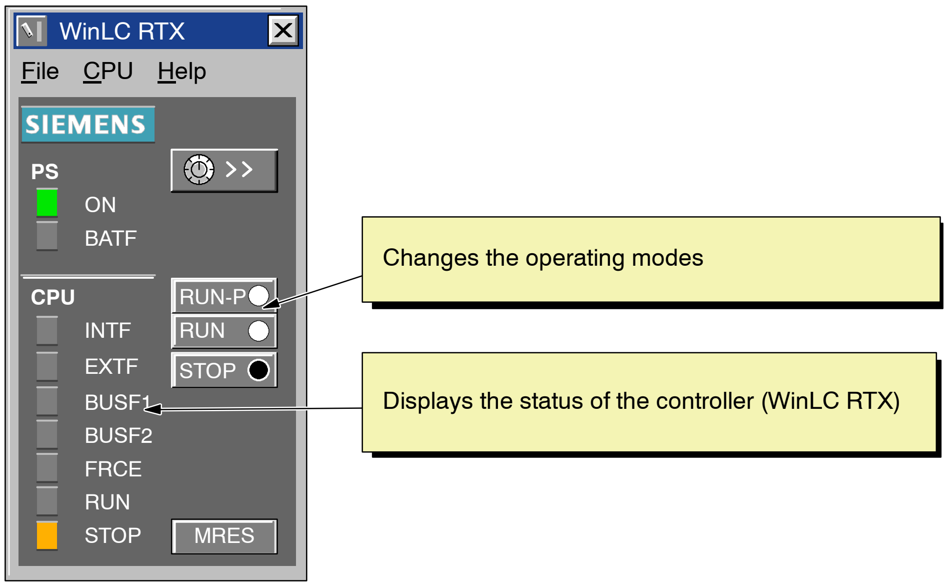


Figure 5-1 Mode Selector Buttons and Status Indicators of the WinLC RTX Control Panel

Selecting the Operating Mode

The RUN, RUN-P, and STOP buttons on the control 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 RTX into the selected operating mode.

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

Table 5-1 Operating Modes of the WinLC RTX Controller

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

Using the Status Indicators

The status indicators on the control panel show basic information about WinLC RTX, such as the current operating mode or the presence of an error condition. Table 5-2 describes the different status indicators for the control panel. You cannot change the status of WinLC RTX 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 RTX.
BATF	Battery fault. Always off for WinLC RTX.
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 RTX supports only one PROFIBUS-DP network, BUSF1 is the only active indicator; BUSF2 is not applicable for WinLC RTX.
FRCE	This indicator lights up (solid) to show that a force request is active. Not applicable for WinLC RTX.
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 RTX 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 RTX controller. 2. Restart the WinLC RTX controller. 3. Reset the memory (MRES). If WinLC RTX is running as a service, you must use the Windows NT control panel to shut down and restart the WinLC RTX controller.

5.2 Resetting the WinLC RTX Memory

The WinLC RTX control 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 RTX memory; however, WinLC RTX must already be in STOP mode.

You normally reset the memory areas before downloading a new program to WinLC RTX or restoring an archive file. You also reset the memory if the STOP indicator on the control 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

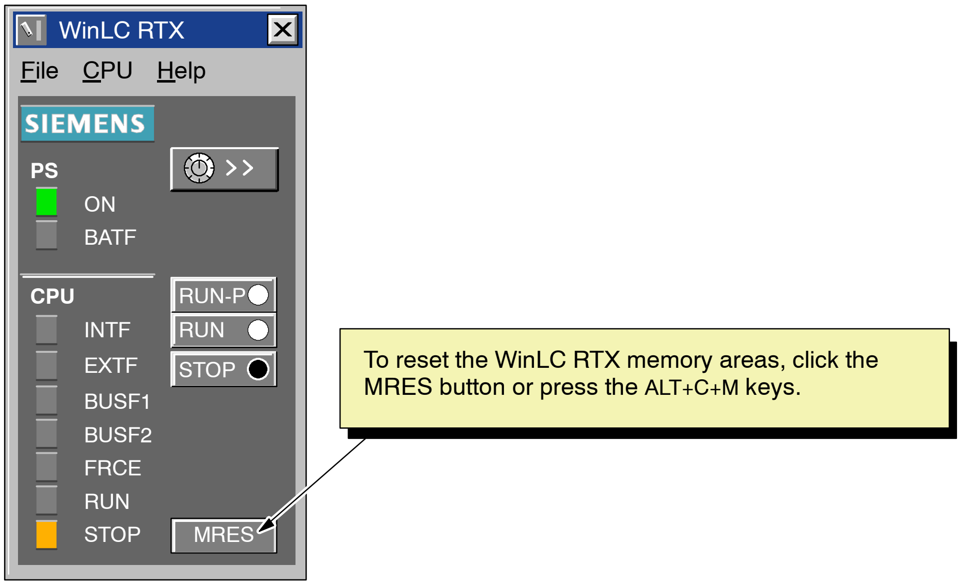
The MRES button performs a memory reset on the memory areas. See Figure 5-2. Clicking the MRES button places WinLC RTX into STOP mode and performs the following tasks:

- WinLC RTX deletes the entire user program from both the work memory area and the load memory area. This includes the data blocks (DBs).
- WinLC RTX 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 RTX memory areas, click the MRES button or press the ALT+C+M keys.

Figure 5-2 Resetting the WinLC RTX Memory with the WinLC Control Panel

5.3 Using the Diagnostic Information Stored in WinLC RTX

As described in Section 5.1, the control panel provides indicators that display information about the status of WinLC RTX. 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 RTX.

Monitoring the Diagnostic Information

When WinLC RTX encounters an error condition while in RUN mode (executing the user program), WinLC RTX 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 RTX 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 RTX, see Section B.2.

WinLC RTX 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 RTX

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 RTX, 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 RTX.
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 RTX. 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 RTX 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 RTX is executing the user program.

5.4 Understanding How WinLC RTX Operates if Windows NT Crashes

WinLC RTX supports OB84 (CPU Hardware Fault), which allows you to initiate the shutdown of your process in case Windows NT detects an unrecoverable fault (”blue screen”) or STOP error while WinLC RTX is running. If WinLC RTX is still able to run after Windows NT has initiated the system shutdown procedure, one of the following occurs:

- If WinLC RTX is in RUN mode and the user program includes OB84, WinLC RTX starts OB84 and continues in RUN mode until the user program calls SFC46 (STP) to place the controller in STOP mode. After WinLC RTX transitions to STOP mode, Windows NT completes its system shutdown.
- If WinLC RTX is in RUN mode and the user program does not include OB84, WinLC RTX transitions to STOP mode and Windows NT completes its system shutdown.
- If WinLC RTX is in STOP mode or if the user program does not include OB84, Windows NT completes its system shutdown.
- If NT is configured to automatically reboot after a STOP error occurs, WinLC RTX automatically restarts if it is configured to run as a service. (To configure automatic reboot, open the System Properties dialog box, select the Startup/Shutdown tab, then check the “automatically reboot” checkbox.)

Note

When WinLC RTX is restarted, it will use the program as it was last downloaded and will execute OB100 if it is present. Event 1382 is used to start OB100.The current/last startup type is shown in the diagnostic buffer as “automatic warm reboot after non-backup power on with system memory reset”. If you want to check for this condition in OB100, see the *System Software for S7-300/400 System and Standard Functions Reference Manual.*

OB100 is always executed after an NT failure, even if OB102 “Cold start” is configured in the STEP 7 Hardware Configuration.

The following restrictions apply:

- The WinLC RTX control panel is unavailable.
- Communication with external systems (such as HMI devices or programming devices) may be unavailable.
- Some system functions may be disabled.
- Cycling the power to the computer initializes all of the program variables to their default values.

5.5 Configuring the Operational Parameters of WinLC RTX

STEP 7 provides the tool for configuring the characteristics and behavior of WinLC RTX. You use the Hardware Configuration tool to display a dialog that configures the operational characteristics for WinLC RTX. 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 WinLC RTX 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 RTX was in STOP mode)

Table 5-4 WinLC RTX Configuration Parameters Provided by STEP 7

Parameters	Description
General	Provides information about WinLC RTX
Startup	Defines the operational characteristics of WinLC RTX 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 RTX 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 RTX 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 RTX 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.

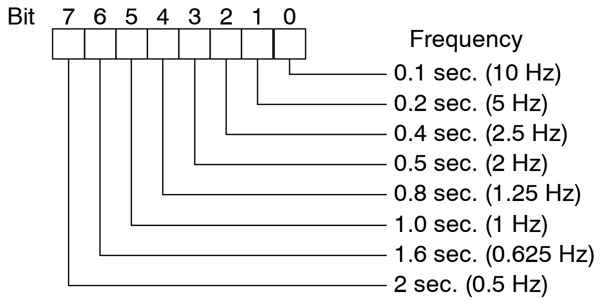


Figure 5-3 Clock Frequencies for the Memory Byte Configured as Clock Memory

Configuring the Scan Cycle

Using the Cycle/Clock Memory tab of the STEP 7 Hardware Configuration, you can configure WinLC RTX 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 and Appendix D.

Note

The minimum cycle time of WinLC RTX encompasses both the time required for executing the user program and the sleep time (which allows your computer to perform other tasks).

WinLC RTX 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 RTX 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 cycle 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 RTX. 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 RTX and 50% could be used by another application (based on process priority).	0 to 6000 ms	0

Using Synchronous I/O Update (Equidistant DP)

With WinLC RTX 3.1, you can operate the DP Master in normal mode or in constant bus cycle time (equidistant) mode. In normal mode, the DP cycle and the PLC cycle operate asynchronously to each other. In constant bus cycle time mode, you can assign a process image partition to the DP master for synchronous update.

In constant bus cycle time mode, the DP cycle begins with a global control command notifying the slaves of the start of the bus cycle, followed by the cyclic I/O update, then acyclic operations, and finally a delay such that the next DP cycle starts on the next multiple of the configured cycle time. During the bus cycle, two events signal the user program:

- At the end of the I/O update, an interrupt schedules the synchronous OB (OB61) for execution.
- At the start of the succeeding cycle (when the global control command is being transmitted to the slaves), an event signals WinLC RTX that further execution of SFC 126 and SFC 127 should return an error.

Between the two events (between the interrupt and the transmission of the global control command), OB61 can call SFC126 and SFC127 to execute synchronous updating of the process image partitions that were assigned to the DP master. If these SFC calls execute without error, the I/O update is synchronized to the process image partition update and occurs at a constant interval between updates.

You can configure the DP bus cycle when you configure network properties for the DP master. To configure the constant bus cycle time mode for WinLC RTX 3.1, you need STEP 7 V5.1 SP3 or higher.

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 RTX 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 RTX 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 RTX 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 RTX. Table 5-10 lists the parameters for the different interrupts.

WinLC RTX 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 RTX 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 RTX 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 RTX scan cycle (see Section 4.6 and Figure 4-10) does not affect the execution of OB35: WinLC RTX 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 RTX 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 RTX 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 RTX (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 RTX clock to synchronize the clocks of other controllers: • None: no synchronization • As Slave: WinLC RTX 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

5.6 System Clock Supported by WinLC RTX

WinLC RTX 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 RTX.

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 RTX application, this difference from the hardware clock is maintained: when you open the WinLC RTX application again, the system clock of the controller reflects the passage of time while the WinLC RTX 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*.

Configuring the PROFIBUS-DP Network 6

Chapter Overview

The WinLC RTX 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 RTX (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*.

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 RTX 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 RTX 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 RTX. A network consists of one or more segments. Each segment can have up to 32 nodes (including repeaters on either end of the segment).

Note

Both WinLC RTX and the communications processor CP 5613 support a total of 125 DP slaves.

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 RTX, WinLC RTX 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 RTX counts as one of these nodes, this means that WinLC RTX can control up to 125 DP 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.

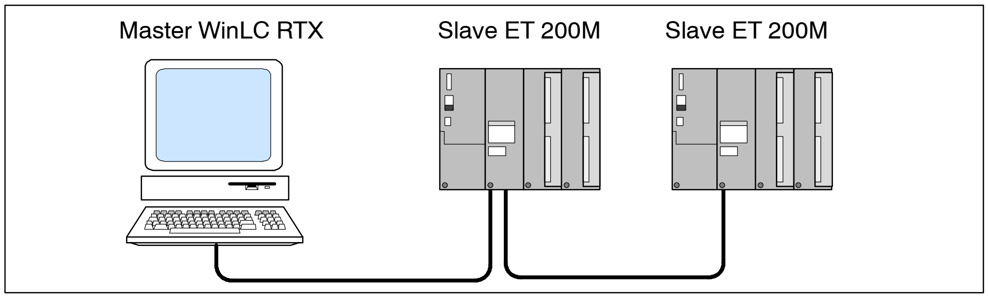


Figure 6-1 Sample PROFIBUS-DP network

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 RTX) 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.

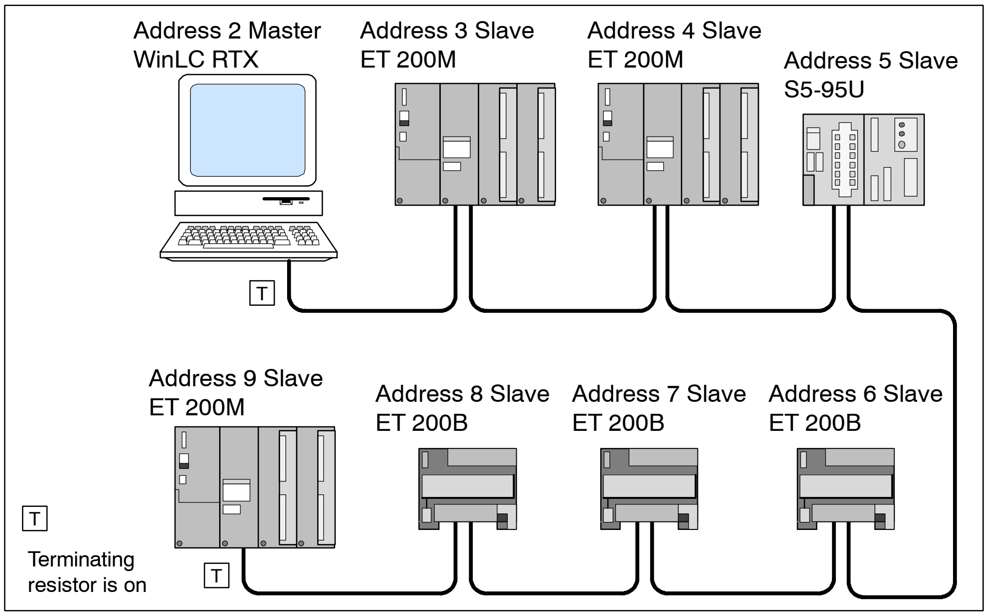


Figure 6-2 Typical Addresses for a PROFIBUS-DP Network

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.

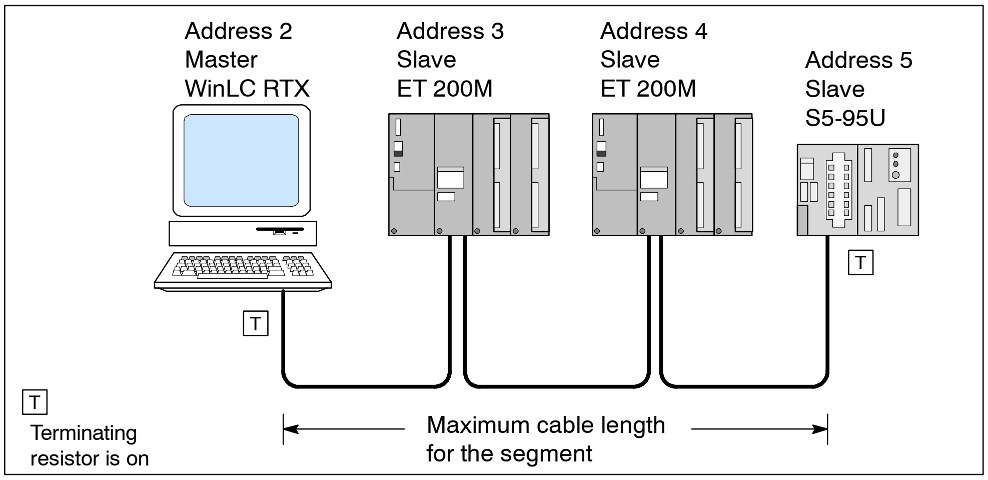


Figure 6-3 Maximum Cable Length for a Segment

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.

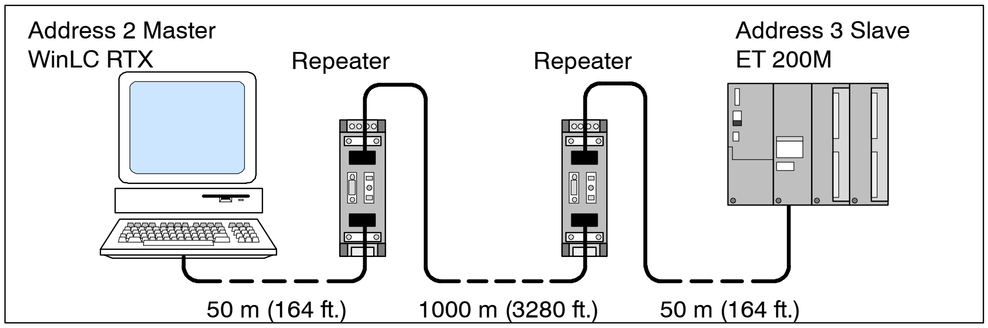


Figure 6-4 Using Repeaters to Increase the PROFIBUS Cable Length

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 RTX by the modules for each node. The PROFIBUS-DP configuration must be downloaded to WinLC RTX 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 use the node address to reference a data. 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 RTX.

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 QUAY1023
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 to 244 bytes	

Specifying the Node Addresses

When you place a DP device into the WinLC RTX configuration using the hardware configuration function of STEP 7, you are prompted to enter the node address for the device. This address identifies the node to the DP master system. The default node address for WinLC RTX is 2. As slaves (I/O modules/racks) are placed on the DP “wire,” STEP 7 displays a default node address. You can change this address as required.

Specifying the Logical I/O Address

During the configuration of WinLC RTX, the I/O data of each module of the PROFIBUS-DP network is allocated a logical address in the input and/or output area. You use these addresses to access the input or output data for the module. Additionally, a base (lowest logical) address for the module is used by WinLC RTX to report module events to the user 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 accessed 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.

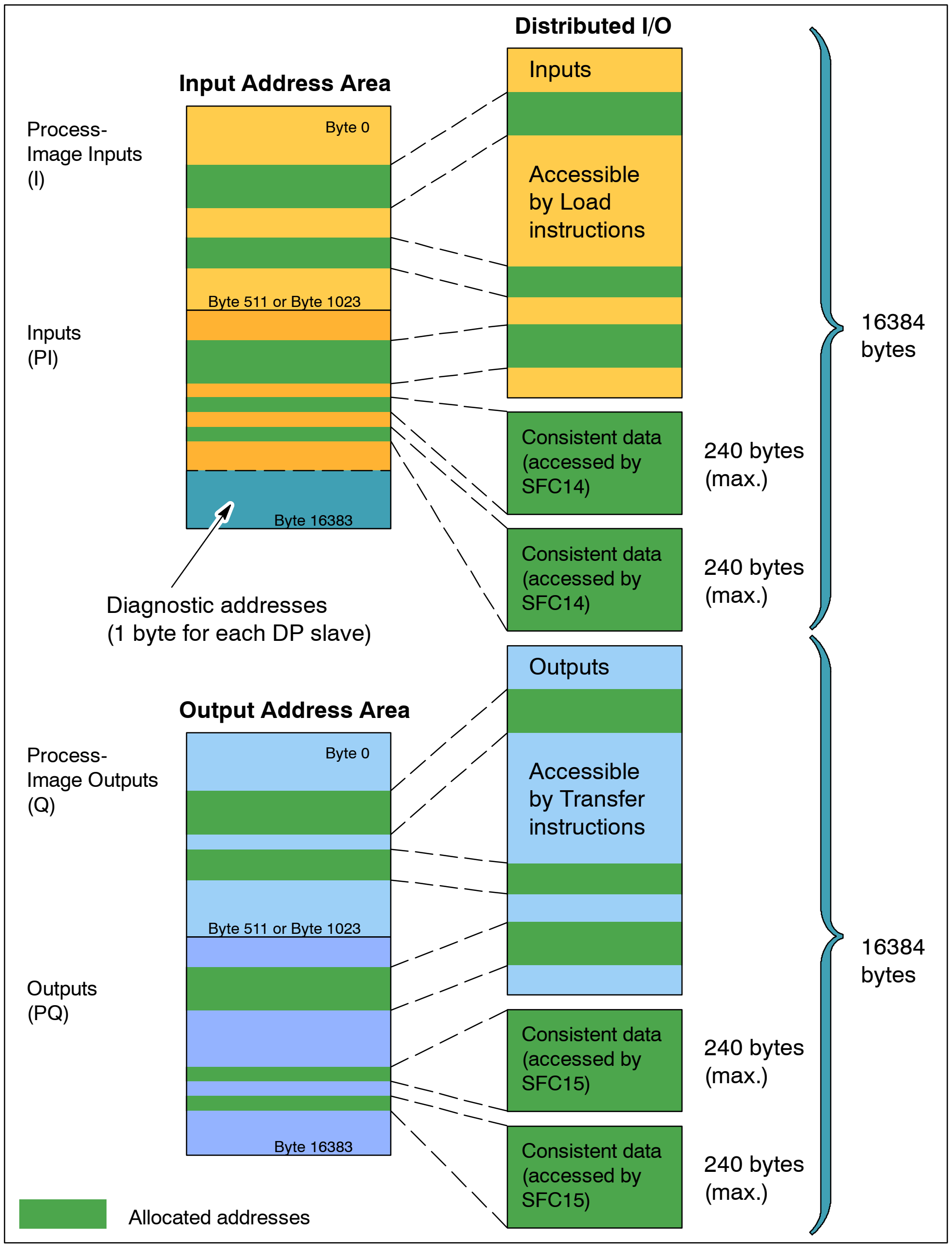


Figure 6-5 Accessing the Distributed I/O

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 RTX, 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). This address is also used by WinLC RTX to report node state changes (in OB86) to the user program.

Note

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

As you use the STEP 7 hardware configuration tools to configure WinLC RTX 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*.

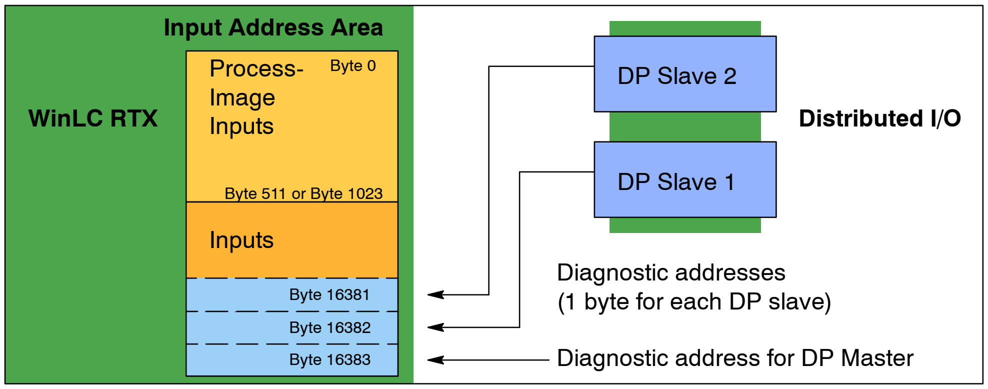


Figure 6-6 Diagnostic Addresses for the Distributed I/O

Troubleshooting for Network Problems

WinLC RTX provides powerful features to help you diagnose DP network problems:

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

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 RTX.
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 RTX (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 RTX 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.

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 RTX 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 RTX, 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 WinLC RTX

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 RTX 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 RTX
		0200	System response
		0300	Language description of WinLC RTX
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 RTX (bytes)	

Table A-1 Sublists of the System Status List (SZL) for WinLC RTX, 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:	0A00 0B00 0C00 0E00 0800	OBs (number and size)
F15	Only SZL partial list header information		
	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		

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

SZL-ID (hexadecimal)	Sublist	Index (hexadecimal)	Record Contents
	Operating status of the CPU		
0124	Last executed operating status transition		
0424	Current operating status	4520	Defective status
0524	Specified operating status	5000	STOP status
		5010	Startup status
		5020	RUN status
		5030	HOLD status
0131	**Communications performance parameters** of the communications type specified	0001	Number of connections and baud rates
		0002	Test and startup parameters
		0003	Operator interface parameters
		0004	Object management system (operating system function)
		0005	Diagnostics functions and diagnostics entries
		0006	PBK: connections; number of send/receive parameters
		0009	Number of run-time meters
0132	**Communications status information** of the communications type specified	0001	Number and type of connections
		0002	Number of test jobs set up
		0003	Operator interface: number of current cyclic read jobs
		0004	Protection levels of WinLC RTX
		0008	Time system, correction factor, run-time meter, date and time of day
		0009	Baud rate (set by means of the MPI)
		000A	Baud rate (set by means of the S7-300 backplane bus)
		0x10	PMC S7 scan: Number of configured messages and time stamp of SDBs for each scan cycle
	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

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

SZL-ID (hexadecimal)	Sublist	Index (hexadecimal)	Record Contents
	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		
	Expanded DP Master system information		Status information regarding synchronicity for the nodes connected to a PROFIBUS-DP network
0095	Information of all DP master systems known to the CPU	0 / DP master sys.-ID | L-Byte = 00H	
0195	Information on a DP master system		
0F95	Header information only		
	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 RTX 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 RTX. In addition to these system blocks, you can use the other S7 blocks to create the user program:

- Function (FC): WinLC RTX supports up to 65,536 FCs (FC0 to FC65535). Each FC can contain up to 65,570 bytes.
- Function block (FB): WinLC RTX supports up to 65,536 FBs (FB0 to FB65535). Each FB can contain up to 65,570 bytes.
- Data block (DB): WinLC RTX 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.

The total number of blocks in the user program that you can download to WinLC RTX is 2500 blocks.

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 RTX is a component of the WinAC RTX package: 6ES7 671-0RC01-0YX0

Features

WinLC RTX provides the following features:

- Accumulators: 4 (ACCU 1 to ACCU 4)
- Communications: PROFIBUS-DP master device
- Work memory and Load memory: limited by the amount of non-paging memory supported by the computer (PC) and the operating system, which is less than the physical memory (RAM) in the computer
- 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 RTX communicates with the distributed I/O as a PROFIBUS-DP master device. As a master device, WinLC RTX can communicate with up to 125 slave devices (either S7-DP slaves or other DP slaves).

Technical Specifications

Table B-1 Technical Specifications of WinLC RTX

WinLC RTX	Description
Work memory Load memory (RAM)	Limited by the amount of non-paging memory of the computer. The following factors affect this amount: • Amount of physical memory (RAM) installed in the computer • Other programs being executed at the same time as WinLC RTX
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 Technical Specifications of WinLC RTX, continued

WinLC RTX	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	 17 (see Table B-2) 7 (see Table B-9) 58 (see Table B-6) 20 FB0 to FB65535 FC0 to FC65535 DB1 to DB65535 (DB0 is reserved) 2500
Nesting depth	24 per OB. Each OB, including the two synchronous OBs (OB121 and OB122), has a nesting depth of 24.
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 • 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 RTX and the user program. Table B-2 lists the OBs which are supported. WinLC RTX 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
OB61	Synchronous cycle interrupt	25 (default)
OB80	Time error	26
OB82	Diagnostic interrupt	24 to 26 (or 28)[1]
OB83	Module remove/insert interrupt	24 to 26 (or 28)[1]
OB84	CPU hardware fault	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 RTX 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 RTX executes OB100 (warm restart) or OB102 (cold restart), based either on the hardware configuration for WinLC RTX or which restart option was selected from a dialog box displayed by the WinLC control panel. After OB100 (or OB102) has been successfully executed, WinLC RTX 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 RTX provides a variety of OBs that interrupt the execution of OB1. Table B-4 lists the different interrupt OBs which are supported by WinLC RTX. 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 The practical minimum time is based on the performance of the computer and the size of the user program.	OB35 OB36	1136_H 1137_H	12 13	↓
Hardware interrupt	OB40	1141_H (channel 1)	16	High

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

- If OB10, OB20, or OB40 is missing and OB85 has not been downloaded, WinLC RTX changes operating mode (from RUN to STOP).
- WinLC RTX remains in RUN mode if OB35 or OB36 is missing or cannot be executed at the specified time.

Note

If you schedule OB35 or OB36 to be executed at a specific interval, make certain that the program can be executed within the time frame and also that your WinLC RTX application can process the OB within the allotted time.

Error OBs

As shown in Table B-5, WinLC RTX 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 RTX 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
CPU hardware fault	OB84	3985_H,	26 (or 28)
Priority class error: • Start event occurs for an OB that has not been downloaded. • During the I/O cycle, WinLC RTX attempts to access a module or DP slave that is defective or not plugged in . • WinLC RTX 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 RTX provides SFCs, which are logic blocks that perform basic tasks. The user program calls the SFC and passes the required parameters; the SFC performs its task and returns the result.

Asynchronous SFCs Supported

WinLC RTX 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, SFC82, SFC83, and SFC84.

Calling Load Memory in Startup

In contrast to the S7-300, WinLC allows both the first call (with REQ = 1) and the second call (with REQ = 0) in STARTUP so the action can be completed in STARTUP.

SFCs That Can Cause the Scan Cycle to Vary

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

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

Execution Times for the SFCs Supported by WinLC RTX

Table B-6 lists the SFCs which are supported.

Table B-6 System Functions (SFCs) Supported

SFC	Name	Description	Execution Time[1]
SFC0	SET_CLK	Sets the system clock	2.82 µs
SFC1	READ_CLK	Reads the system clock	0.16 µs
SFC2	SET_RTM	Sets the run-time meter	-0.34 µs
SFC3	CTRL_RTM	Starts or stops the run-time meter	2.85 µs
SFC4	READ_RTM	Reads the run-time meter	-0.34 µs
SFC5	GADR_LGC	Queries the logical address of a channel	2.16 µs
SFC6	RD_SINFO	Reads the start information of an OB	2.51 µs
SFC11	DPSYNC_FR	Synchronize groups of DP slaves	-0.34 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	-0.03 µs
SFC14	DPRD_DAT	Reads the consistent data from a DP slave	-0.34 µs

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

SFC	Name	Description	Execution Time[1]
SFC15	DPWR_DAT	Writes the consistent data to a DP slave	–0.34 µs
SFC17	ALARM_SQ	Generates an acknowledgeable block-related message	4.87 µs
SFC18	ALARM_S	Generates an unacknowledgeable block-related message	4.76 µs
SFC19	ALARM_SC	Queries the status for the last message (SFC17 or SFC18)	1.17 µs
SFC20	BLKMOVB	Copies variables	2.67 µs
SFC21	FILL	Initializes a memory area 1 word 50 words 100 words	2.66 µs –0.34 µs –0.34 µs
SFC22	CREAT_DB	Creates a data block in work memory	8.12 µs
SFC23	DEL_DB	Deletes a data block. WinLC RTX 3.1 allows an application to delete a non-sequence relevant data block.	2.70 µs
SFC24	TEST_DB	Provides information about a data block. In WinLC RTX 3.1, SFC24 can return DB length and write protection flags for non-sequence relevant data blocks, although it returns error code 80B2 for non-sequence relevant data blocks.	1.17 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	–0.17 µs
SFC27	UPDAT_PO	Updates the process-image output table	16.33 µs
SFC28	SET_TINT	Sets the time-of-day interrupt (OB10)	3.39 µs
SFC29	CAN_TINT	Cancels the time-of-day interrupt (OB10)	2.19 µs
SFC30	ACT_TINT	Activates the time-of-day interrupt (OB10)	1.47 µs
SFC31	QRY_TINT	Queries the time-of-day interrupt (OB10)	1.66 µs
SFC32	SRT_DINT	Starts the time-delay interrupt (OB20)	5.93 µs
SFC33	CAN_DINT	Cancels the time-delay interrupt (OB20)	1.90 µs
SFC34	QRY_DINT	Queries the time-delay interrupt (OB20)	1.67 µs
SFC36	MSK_FLT	Mask synchronous errors	–0.34 µs
SFC37	DMSK_FLT	Unmask synchronous errors	–0.34 µs
SFC38	READ_ERR	Read the error register	–0.34 µs
SFC39	DIS_IRT	Disables the processing of all new interrupts	1.00 µs
SFC40	EN_IRT	Enables the processing of new interrupts	1.33 µs
SFC41	DIS_AIRT	Disables the processing of new interrupts with higher priority than the current OB	0.67 µs
SFC42	EN_AIRT	Enables the processing of new interrupts with higher priority than the current OB	2.00 µs
SFC43	RE_TRIGR	Retriggers the watchdog timer (monitoring the cycle time)	6.65 µs
SFC44	REPL_VAL	Transfers a value to ACCU1 (accumulator 1)	20.58 µs
SFC46	STP	Changes the operating mode to STOP	Not applicable
SFC47	WAIT	Delays the execution of the user program	496.15 µs
SFC49	LGC_GADR	Queries the module slot belonging to a logical address	1.66 µs
SFC50	RD_LGADR	Queries all of the logical addresses of a module	4.19 µs
SFC51	RDSYSST	Reads all or part of a system status list	4.71 µs

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

SFC	Name	Description	Execution Time[1]
SFC52	WR_UMSG	Writes a user element to the diagnostics buffer	8.70 µs
SFC54	RD_PARM	Read the defined parameter	-0.67 µs
SFC55	WR_PARM	Write the defined parameter	-0.67 µs
SFC56	WR_DPARM	Write the default parameter	-0.67 µs
SFC57	PARM_MOD	Assign the parameters to a module	-0.67 µs
SFC58	WR_REC	Write a data record	-0.67 µs
SFC59	RD_REC	Read a data record	-0.67 µs
SFC64	TIME_TCK	Reads the time from the system clock	0.67 µs
SFC78	OB_RT	Reports OB run-time information	Not applicable
SFC79	SET	Set a range of outputs	-0.34 µs
SFC80	RESET	Reset a range of outputs	-0.34 µs
SFC82	CREA_DBL	Creates a block in load memory	7.00 µs
SFC83	READ_DBL	Copies data from a block in load memory	5.00 µs
SFC84	WRIT_DBL	Writes to Load Memory blocks so data is saved immediately. Load memory blocks that are used to rcover from an abnornal termination can be updated while the program is running. Use SFC84 only for larger segments of a database, not for frequent variable processing.	6.00 µs
SFC126 SFC127	SYNC_PI SYNC_PO	Synchronous update of process image partition of inputs Synchronous update of process image partition of outputs	24.00 µs 37.00 µs

1 The execution times were measured on a Dell Dual 1G computer. Tuning settings: 9000 µs execution time limit; 90% maximum execution load; 1000 µs forced execution sleep. Actual execution times may vary, depending on your computer.

OB Runtime Statistics (SFC78)

You can use SFC 78 (OB_RT) to capture the following run-time information:

- Priority assigned to an OB
- Total execution time of the previous execution of an OB
- Total elapsed time of the previous execution of an OB
- Time of the start event of an executing OB
- Actual execution time of an executing OB
- Elapsed time of an executing OB

You can use the information from calling SFC 78 to profile the intervals between successive executions of an OB, to profile the total execution or startup delay times of an OB, and other ways. WinLC RTX does not support using SFC 78 to determine the current delay in starting a pending OB execution.

Error Codes for Load Memory SFCs (SFC82, SFC83, SFC84)

WinLC RTX returns the following error codes for the RET_VAL parameter:

Table B-7 Load Memory Error Codes (SFC82, SFC83, SFC84)

Error Code	Message
80BB	Write to hard disk failed
8092	Program calls SFC 82 during a Windows operating system failure (”blue screen”)
80C3	WinLC limit of 32 outstanding SFC 82 – 84 jobs has been exceeded
8022	Source of type BOOLEAN does not have repetition factor divisible by 8
8023	Destination of type BOOLEAN does not have repetition factor divisible by 8
8024	Source of type STRING does not have repetition factor equal to 1
8025	Destination of type STRING does not have repetition factor equal to 1
8093	Destination (SFC 83) or source (SFC 84) is not sequence relevant (Unlinked = 0, false)
80B1	Source data block is not in Load Memory
80B4	Destination block has an F attribute
8094	NON_RETENTIVE attribute bit is not supported

Note

WinLC returns an error code 8092 if SFC 82, SFC 83, or SFC84 is called after a Windows NT failure. Applications that need to continue operating after a Windows NT failure should check for this error code.

B.4 Execution Times of the DP Instructions

Table B-8 lists the execution times for the SFCs used with the distributed I/O.

Table B-8 Execution Times of the DP Instructions

SFC	Name	Description	CP5613[1]
SFC11	DPSYNC_FR	Synchronize groups of DP slaves	-0.34 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	-0.03 µs
SFC14	DPRD_DAT	Reads the consistent data from a DP slave	-0.34 µs
SFC15	DPWR_DAT	Writes the consistent data to a DP slave	-0.34 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	-0.17 µs
SFC27	UPDAT_PO	Updates the process-image output table	16.33 µs

1 The execution times were measured on a Dell Dual 1G computer. Tuning settings: 9000 µs execution time limit; 90% maximum execution load; 1000 µs forced execution sleep. Actual execution times may vary, depending on your computer.

B.5 System Function Blocks (SFBs) Supported

WinLC RTX provides SFBs, which are logic blocks similar to SFCs. Table B-9 lists the SFBs which are supported. When the user program calls an SFB, a data block (DB) must also be assigned.

Table B-9 System Function Blocks (SFBs) Supported

SFB	Name	Description	Execution Time[1]
SFB0	CTU	Provides a "count up" timer	1.70 µs
SFB1	CTD	Provides a "count down" timer	1.68 µs
SFB2	CTUD	Provides a "count up/down" timer	2.16 µs
SFB3	TP	Generates a pulse	1.66 µs
SFB4	TON	Generates an on-delay timer	1.66 µs
SFB5	TOF	Generates an off-delay timer	1.66 µs
SFB32	DRUM	Implements a sequencer	-0.34 µs

1 The execution times were measured on a Dell Dual 1G computer. Tuning settings: 9000 µs execution time limit; 90% maximum execution load; 1000 µs forced execution sleep. Actual execution times may vary, depending on your computer.

B.6 Execution Times of Instructions

The execution times listed in Table B-10 (execution times for math operations) and Table B-11 (execution times for instructions) reflect the average execution times for STEP 7 programs running on WinLC RTX. Actual execution times may vary, depending on your system.

Table B-10 Execution Times of Math Operations (µs)

Math Operation	Integer[1]	Real[1]	Double Word[1]
Addition (+)	0.03 µs	0.03 µs	0.03 µs
Subtraction (-)	0.03 µs	0.03 µs	0.03 µs
Multiplication (*)	0.00 µs	0.03 µs	0.07 µs
Division (/)	0.07 µs	0.07 µs	0.07 µs

1 The execution times were measured on a Dell Dual 1G computer. Tuning settings: 9000 µs execution time limit; 90% maximum execution load; 1000 µs forced execution sleep. Actual execution times may vary, depending on your computer.

Table B-11 Execution Times (µs) per Instruction

Instructions			Execution Time[1]	
			Direct addressing	**Indirect addressing**
Boolean operations: A, AN O, ON, X, XN	Memory areas:	I	0.05 µs	0.05 µs
		M	0.03 µs	0.07 µs
		L	0.05 µs	0.05 µs
		DB	0.05 µs	0.05 µs
		T	0.08 µs	0.12 µs
		C	0.03 µs	0.07 µs
Boolean operations (on the accumulator): ==I, <>I, >I, <I, >=I, <=I			0.06 µs	
Operations on the bits of the status word: A==0, A<>0, A>0, A<0, A>=0, A<=0			0.06 µs	
Transitional contacts:	Edge Positive	FP	-0.22 µs	
	Edge Negative	FN	-0.22 µs	
Set/Reset operations	Set	S	0.12 µs	0.08 µs
(bit operands)	Reset	R	0.12 µs	0.08 µs
RLO Operations	Negate RLO	NOT	0.03 µs	
	Set RLO	SET	0.05 µs	
	Clear RLO	CLR	0.02 µs	
	Save RLO	SAVE	0.03 µs	
Operations on Timers	Pulse timer	SP	0.13 µs	0.15 µs
	Reset (timer)	R	0.05 µs	0.07 µs
	Extended pulse timer	SE	0.13 µs	0.15 µs
	On-delay timer	SD	0.13 µs	0.17 µs
	Retentive on-delay timer	SS	0.13 µs	0.17 µs
	Off-delay timer	SF	0.17 µs	0.20 µs
Miscellaneous:	Open DB	OPN	0.33 µs	
	Load	L	0.16 µs	
	Transfer	T	0.17 µs	
	SFC126	SYNC_PI	24.00 µs	
	SFC127	SYNC_PO	34.00 µs	

[1] The execution times were measured on a Dell Dual 1G computer. Tuning settings: 9000 µs execution time limit; 90% maximum execution load; 1000 µs forced execution sleep. Actual execution times may vary, depending on your system.

Panel Control C

Chapter Overview

The WinLC control panel is also available as an ActiveX component for use in SIMATIC Computing. The Panel control permits access from the SoftContainer provided by SIMATIC Computing or from any other ActiveX container.

The Panel control provides access to the operating modes of either WinLC (WinAC RTX) 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.

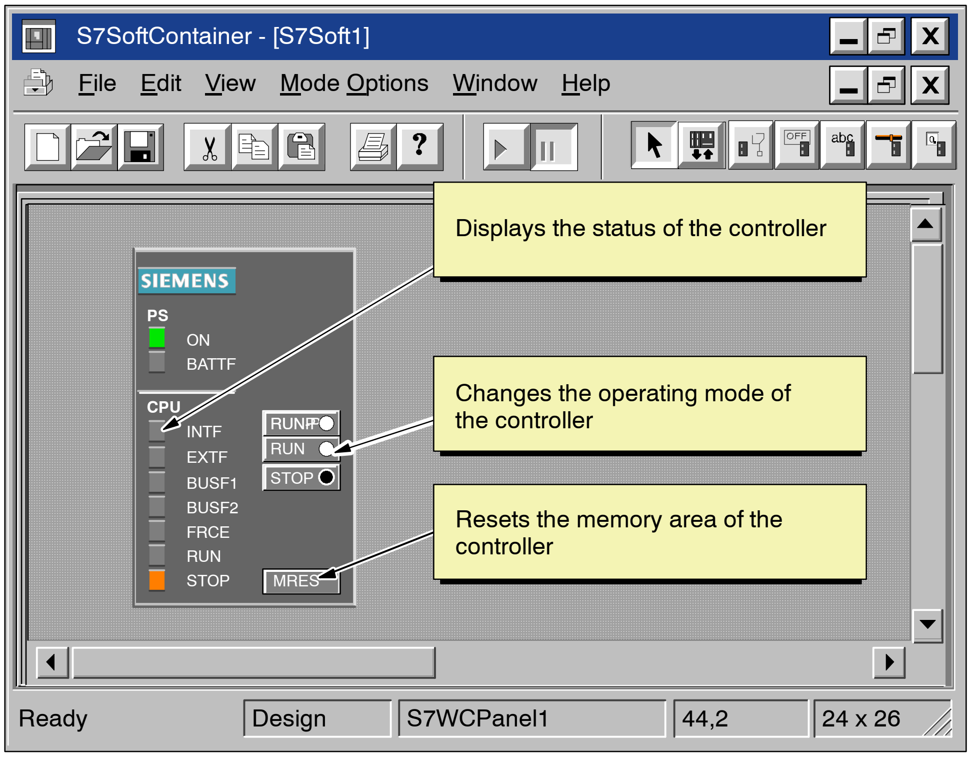

S7SoftContainer - [S7Soft1]
File
Edit
View
Mode Options
Window
Help
Displays the status of the controller
SIEMENS
PS
ON
BATTF
CPU
INTF
EXTF
BUSF1
BUSF2
FRCE
RUN
STOP
RUN-P
RUN
STOP
MRES
Changes the operating mode of the controller
Resets the memory area of the controller
Ready
Design
S7WCPanel1
44,2
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 RTX.
BATTF	Battery fault. Always off for WinLC RTX.
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 RTX supports only 1 PROFIBUS-DP network, BUSF1 is the only active indicator; BUSF2 is not applicable for WinLC RTX.
FRCE	This indicator lights up (solid) to show that a force request is active. Not applicable for WinLC RTX.
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 RTX 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 RTX controller. 2. Restart the WinLC RTX controller. 3. Reset the memory (MRES). If WinLC RTX is running as a service, you must use the Windows NT control panel to shut down and restart the WinLC RTX 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:

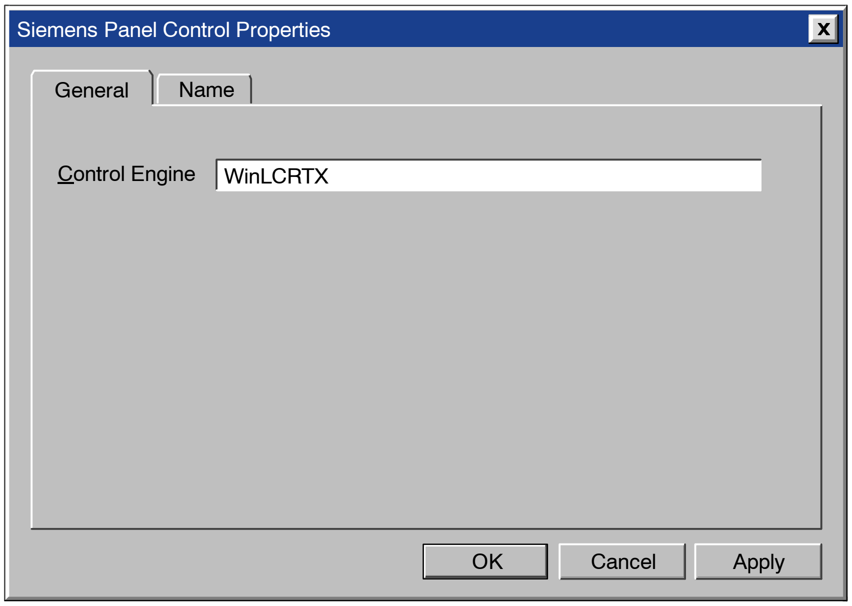
1. The controller deletes the entire user program, including the data blocks (DBs).
2. The controller resets the memory areas (I, Q, M, T, and C).

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:

- For WinAC RTX:
  - **`WinLCRTX`** (for WinLC RTX)
- For WinAC Pro:
  - **`CPU 412-2 PCI`** (for the PCI version of the S7 CPU 412)
  - **`CPU 416-2 PCI`** (for the PCI version of the S7 CPU 416)
  - **`CPU 416-2 DP ISA`** (for the ISA version of the S7 CPU 416)



Control Engine: WinLCRTX

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 RTX. This property is valid only for WinLC RTX. The Autostart feature allows you to start WinLC RTX back up in the same operating mode (STOP, RUN, or RUN-P) that it was in before it was shut down.

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 RTX is enabled.
False	(default) The autostart feature of WinLC RTX 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 RTX 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()`**

WinLC RTX and PC Time Sharing D

Chapter Overview

This appendix explains the elements of WinLC RTX control program execution and how WinLC RTX shares PC time with WinLC RTX communication and other PC applications.

D.1 Time Sharing and Control Program Execution

Figure D-1 shows execution activities in three operating system priority categories: Interrupts and Higher Priority Applications, Control Program Execution, and Communication and Lower Priority Applications.

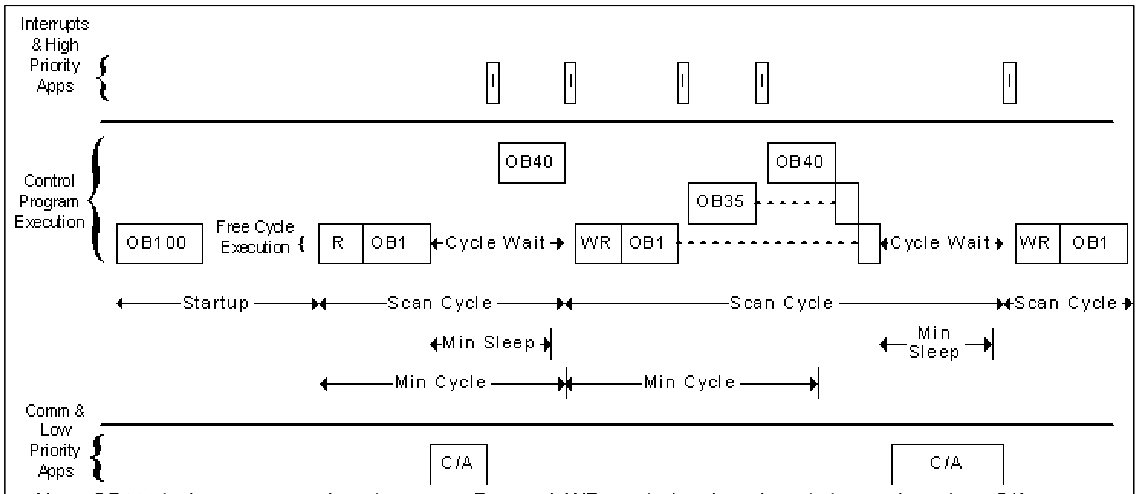
Interrupts and Higher Priority Applications includes all software that has a priority higher than control program execution. Any activity in this group interrupts control program execution at the time the operating system has scheduled the activity to begin execution. Activities in this category include the following:

- Timer events that start free cycle (OB1), cyclic (OB3x) and time-of-day interrupt (OB1x), time-delay (OB2x) OBs and any OB delayed using SFC 47“WAIT.”
- Profibus CP card driver events such as diagnostic events and equidistant I/O events.
- Other control events such as events that cause a transition back to CPU STOP mode.
- Any application or driver with priority greater than the priority set in the WinLC RTX Tuning display.

Control Program Execution is the activity that executes all blocks in the control program. WinLC RTX manages the relative priority of each OB, swapping execution from one OB to another. All OBs are executed at the same Windows or RTX operating system priority.

You can configure other applications to have the same priority as control program execution. The result on control program behavior is similar to an Interrupt activity, that is, the application affects the deterministic behavior of control program execution.

Communication and Lower Priority Applications includes WinLC RTX communication with other applications and all applications with a priority lower than Control Program Execution.



Note: OB100 is the start event, I are interrupts, R = read, WR = write/read, cycle wait time = sleep time, C/A = communications and applications programs.

Figure D-1 WinLC RTX Control Program Execution

D.2 Real-Time Execution Priority

Operating systems such as Windows 9x/NT/2000/XP/CE and VenturCom RTX implement a concept of execution threads (or tasks). Each application has one or more threads and each thread has a priority. The operating system executes threads with the highest priority first. It only executes a lower priority thread when all the higher priority threads are suspended (for example, to wait for some other activity to complete or to “sleep” for a time). Figure D-2 illustrates the thread priority concept and how it relates to the WinLC RTX control program execution thread.

Note

Windows and VenturCom RTX support PCs with more than one CPU. These are called Symmetric Multi-Processing (SMP) machines. A common configuration has two CPUs. SMP machines allow each CPU to execute a thread. When WinLC RTX is executing on an SMP machine, control program execution has a less significant effect on other PC activities.

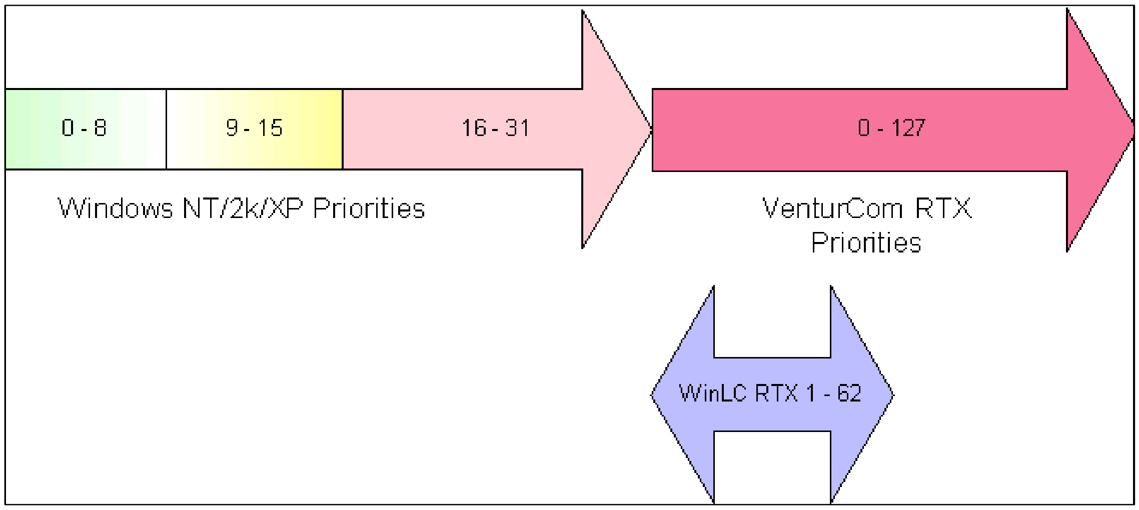


Figure D-2 Windows and VenturCom RTX Priorities

To use WinLC RTX, you need to know about the VenturCom RTX priority scheme. WinLC RTX includes a tuning panel (Figure D-3) that you use to set the WinLC RTX control program execution priority.

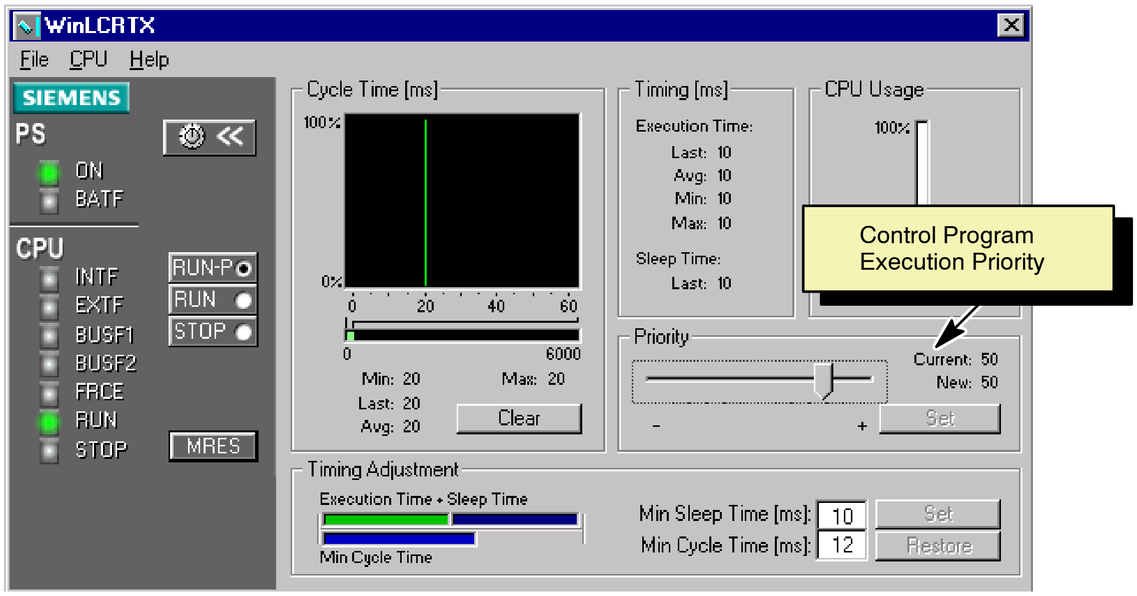


Figure D-3 Tuning Panel, WinLC RTX Control Program Execution Priority

---

Note

When you change the priority using the tuning panel, WinLC RTX automatically ensures that its interrupt activities, such as those which schedule interrupt OBs, are also set to an appropriate priority (see Figure D-1, WinLC RTX Control Program Execution). However, WinLC RTX does not control priorities in customer software, such as asynchronous threads in WinLC RTX Open Development Kit (ODK) software, or other applications in the same environment.

For simplicity, the Windows priorities can be divided into three categories.

- Priority 8 and below are the “Normal and Background” priorities. Normal priority (priority 8) is the default that most applications use.
- Priorities from 9 to 15 are “Above Normal” priorities.
- Priority 16 begins the “Soft real-time” priority range.
- WinLC RTX executes in the “hard real-time” RTX priority range that is higher than all the Windows priorities.

VenturCom provides a Real-Time Extension (RTX) product for Windows operating systems. VenturCom RTX implements a Real-Time Subsystem (RTSS) environment for applications that need both real-time determinism and efficient interoperation with other Windows applications on the same hardware platform.

WinLC RTX executes in the VenturCom Real-time Subsystem (RTSS) environment. Application threads in the RTSS execute at a higher priority than any Windows software unless the application makes a function call to the Windows operating system. The WinLC RTX control program is only affected by higher priority WinLC RTX events, other RTSS applications and drivers, and the hardware platform.

WinLC RTX has the most deterministic behavior with scan cycle jitter less than 1 millisecond. WinLC RTX should be used for the most demanding control applications.

Since all priorities in the RTSS environment are higher priority than any Windows activity, the default installation priority of 50 is usually satisfactory. When other RTSS applications are running, you must determine whether to set the WinLC RTX program execution priority above or below the priority of these applications. Set the WinLC RTX execution priority lower than applications such as drivers, but higher than background applications.

When WinLC RTX is executing at a higher priority, the execution sleep times determine how much time is given to other PC activities.

D.3 Control Program Sleep Management

The previous sections explain that WinLC RTX control program execution must be stopped (or sleep) periodically for all WinLC RTX priority settings at or above Windows normal priority (priority 8). At the real-time priorities, this sleep time must occur every 50 milliseconds or less to allow Windows mouse movement and other Windows applications to operate smoothly.

WinLC RTX implements three mechanisms for sleep management.

- The first mechanism is based on the scan cycle (OB1 execution and the I/O image updates).
- The second mechanism monitors the amount of sleep time that occurs in a “Wake” interval and forces some execution sleep time as needed.
- The third mechanism is under application control and makes use of the “WAIT” system function call (SFC 47).

Free Cycle Sleep Management

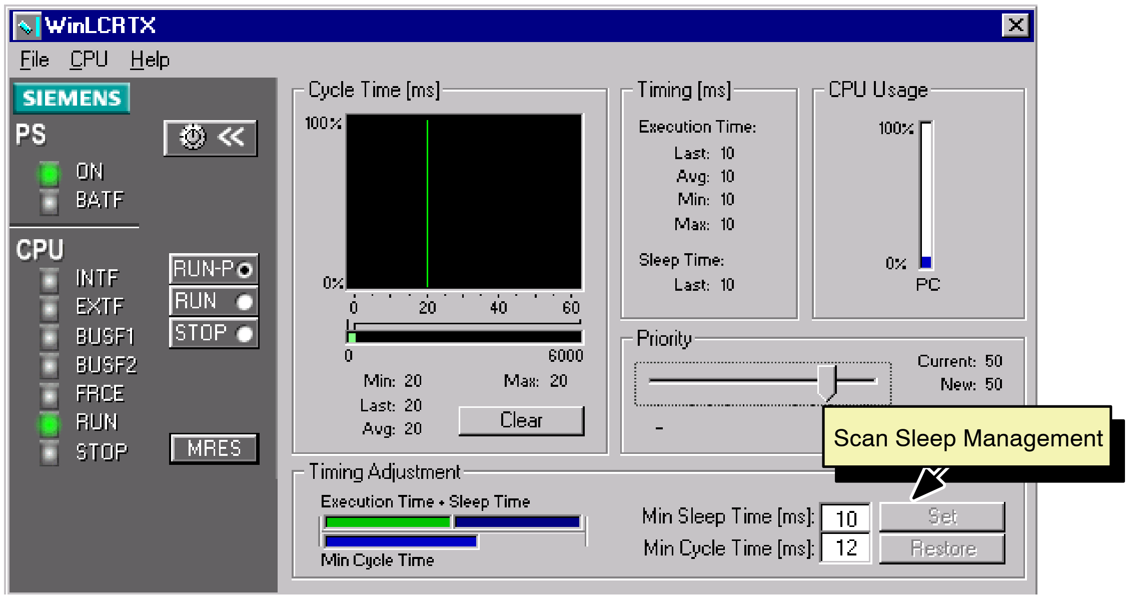
Sleep time (Free cycle wait time) begins when an execution of OB1 completes. It is the time between the completion of OB1 and the start of the next free cycle. The default minimum time between scans is 10 milliseconds. You can change the free cycle wait (or sleep) time using the Tuning panel.

Note

Sleep time (free cycle wait time) only applies to OB1 execution. If a higher priority OB is scheduled to run during this time, it interrupts the free cycle wait interval to execute the higher priority OB so that wait time for the free cycle does not delay the handling of OB interrupts. The consequence of this interruptible wait time is that it is possible to starve the PC of CPU time in control programs that execute many high priority OBs for a period of time.

- The “Min Sleep Time” parameter is the minimum number of milliseconds between the completion of OB1 and the start of the next free cycle.
- The “Min Cycle Time” parameter sets the minimum number of milliseconds from the start of one cycle to the start of the next cycle. This value must be greater than the execution time of the scan before it causes any sleep time to occur between scans.

The difference between these two parameters (minimum sleep time and minimum cycle time) is that the minimum sleep time by itself results in a **fixed sleep time** and a **variable scan time**, depending on how long it takes to execute OB1 and do the process image updates. Conversely, the minimum cycle time results in a **variable sleep time** and a **fixed scan time** when the minimum cycle time is large enough.



Cycle Time [ms]: Min: 20 Max: 20 Last: 20 Avg: 20

Timing [ms]: Execution Time: Last: 10 Avg: 10 Min: 10 Max: 10; Sleep Time: Last: 10

Priority: Current: 50 New: 50

Min Sleep Time [ms]: 10
Min Cycle Time [ms]: 12

Figure D-4 Tuning Panel, Scan Sleep Management

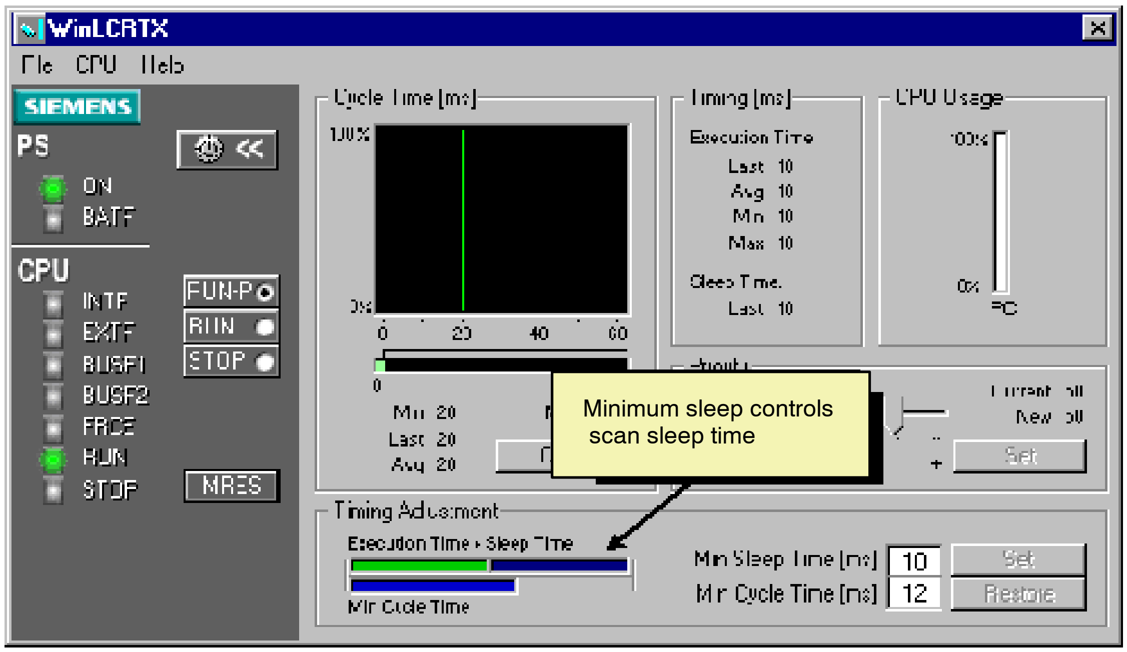
Note

Do not set the minimum scan cycle time to be longer than the scan cycle monitoring time (the watchdog time) configured in the STEP 7 Hardware Configuration Editor. If you set the minimum scan time to a value larger than the watchdog time, WinLC RTX goes to STOP mode during the first scan at the end of the watchdog time interval.

There is some interaction between the minimum sleep time and the minimum scan time parameters. The minimum sleep time guarantees the configured amount of sleep time between each scan, even if the minimum cycle time is too small. The actual sleep time is either the minimum sleep time or the computed sleep time from the minimum cycle time parameter, whichever is larger. The bar graphs to the left of the parameter entry fields show which sleep parameter is controlling the free cycle sleep time.

Figure D-5 shows a case where the minimum sleep time determines the scan cycle sleep time at cycle control point. This example is also shown as the first cycle in Figure D-1, WinLC RTX Control Program Execution.

- Scan execution time = 10ms
- Min Sleep = 10ms
- Min Scan = 12ms implies sleep time = 2ms (12ms – 10ms execute time).
- Sleep time = maximum of 10ms and 2ms = 10ms



Timing [ms]
Execution Time
Last 10
Avg 10
Min 10
Max 10
Sleep Time
Last 10

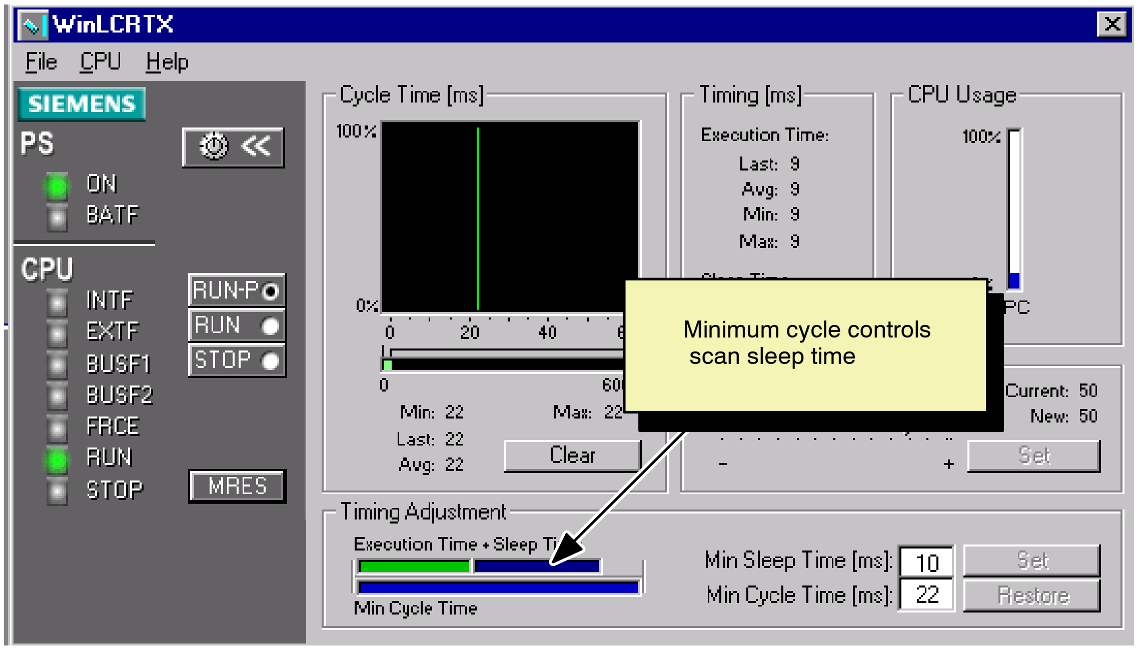
Cycle Time [ms]
Min 20
Last 20
Avg 20

Min Sleep Time [ms] 10
Min Cycle Time [ms] 12

Figure D-5 Configuring Sleep Time, Example 1

Figure D-6 shows a case where the minimum scan time determines the scan cycle sleep time at cycle control point. This example is also shown as the second cycle in Figure D-1.

- Scan execution time = 10ms
- Min Sleep = 10ms
- Min Scan = 22ms implies sleep time = 12ms (22ms – 10ms execute time).
- Sleep time = maximum of 10ms and 12ms = 12ms



Min: 22 Max: 22
Last: 22
Avg: 22

Execution Time:
Last: 9
Avg: 9
Min: 9
Max: 9

Current: 50
New: 50

Min Sleep Time [ms]: 10
Min Cycle Time [ms]: 22

Figure D-6 Configuring Sleep Time, Example 2

The following situations can increase the free cycle time:

- WinLC RTX executes other OBs (such as OB20 and OB35) with higher priorities than OB1.
- Another RTX application that is running on your computer has a higher priority.
- You use a variable table (VAT) with the STEP 7 programming software to display the status of the user program.
- 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 RTX.

You can use the tuning panel CPU Usage bar to see the effect of the sleep parameter setting on CPU load. Note that when WinLC RTX is installed, the CPU Usage shows the percentage of CPU time used that is available to Windows. It does not show the CPU time used in the RTX real-time subsystem.

Control Program Sleep Monitor

Although for many applications, the free cycle sleep management is sufficient to ensure that a WinLC RTX control program does not starve other PC activities of CPU time, there are some cases where you must insert additional sleep time into the control program execution.

Some situations when the free cycle sleep management is not sufficient are:

- The STARTUP organization block (e.g. OB100) takes longer than 50 milliseconds. This can happen by accident, because of a loop in the control program logic, or because the initialization logic takes a long time to execute. S7 PLCs, including WinLC RTX, are required to turn the watchdog timer off during startup. So, if a long startup time is caused by a program error, you cannot debug the control program and might have to reboot the PC.
- A RUN organization block (for example, OB1) takes more than 50 milliseconds to execute. The sleep time described in Section D.3 is applied at the end of OB1. Whenever OB1 takes more than 50 milliseconds, the behavior of other PC applications is noticeably affected because the time given to Windows is spaced too far apart.
- An application has many interrupt OBs (for example, OB35, OB61, OB40) that are not affected by the free cycle wait (sleep) time. With these interrupts, it is possible to write applications (accidentally or on purpose) that use most or all of the free cycle wait time to execute the interrupt OBs. In these cases, the free cycle wait time does not serve its purpose of giving up the CPU for other PC activities.

The WinLC RTX sleep-monitoring algorithm forces control program execution (all OBs) to sleep for a short period of time when one of the conditions described above occurs.

The sleep-monitoring algorithm uses two parameters, a wake interval and a sleep interval. During the wake interval, the algorithm monitors the actual amount of time that no OBs are executing (control program sleep time).

- If the measured sleep time is greater than the sleep interval time, the algorithm does nothing and starts another wake interval.
- If the measured sleep time is less than the sleep interval time, all OBs are stopped from executing for the remainder of the required sleep interval time.

Any control program sleep time imposed because of the sleep-monitoring algorithm is subtracted from the sleep time configured for the end of the free cycle, using the minimum sleep time parameters described in “How Control Sleep Monitoring Affects Your Program.”

The default value for the wake interval is 9 milliseconds; the default value for the sleep interval is 1 millisecond. This ratio ensures that WinLC RTX control program execution cannot use more than 90% of the CPU time in any of the worst case situations described above.

The following examples illustrate how the sleep-monitoring algorithm can affect control program execution time.

How Control Program Sleep Monitoring Affects Your Program

If your control program executes with a scan cycle time, including sleep time, less than the wake interval time (9 milliseconds default), then the sleep-monitoring algorithm should not affect your control program at all. Otherwise, your program is stopped periodically for up to the sleep interval time (1 millisecond default). This has the following consequences:

- An OB is stopped from executing in the middle of the OB logic. This means that the lapse time from start to end of the OB is greater than the actual execution time of the OB.
- The start of an interrupt OB (for example, OB35, OB61, OB40) can be delayed by the amount of the sleep interval time (1 millisecond default). This delay appears as a “jitter” or latency in the actual start time of the OB.

The next sections illustrate these two consequences using example applications.

Example - Sleep-Monitoring Extending OB Completion Time

Consider a control program that checks a one-second timer in a loop in OB1 so that OB1 always takes one second of clock time, or lapse time, to complete.

Table D-1 WinLC RTX Parameters

WinLC RTX Configuration	Value
Number of CPUs	1
Tuning Panel Priority	50 (default WinLC RTX)
Tuning Panel Min Sleep Time	0 milliseconds
Tuning Panel Min Scan Time	0 milliseconds (default)
Sleep-Monitoring Wake Interval	9 milliseconds (default)
Sleep-Monitoring Sleep Interval	1 millisecond (default)

In this case, all PC applications, including the control panel, would be locked up (starved of CPU time) without the control program sleep-monitoring algorithm. The control program would be executing, but the PC would have to be rebooted before other PC applications could be used.

With the sleep-monitor algorithm functioning, OB1 is suspended every 9 milliseconds for one millisecond. This means that when OB1 completes after 1 second of clock time, it has spent 900 milliseconds executing the control program and 100 milliseconds sleeping.

Figure D-7 shows how the tuning panel would look.

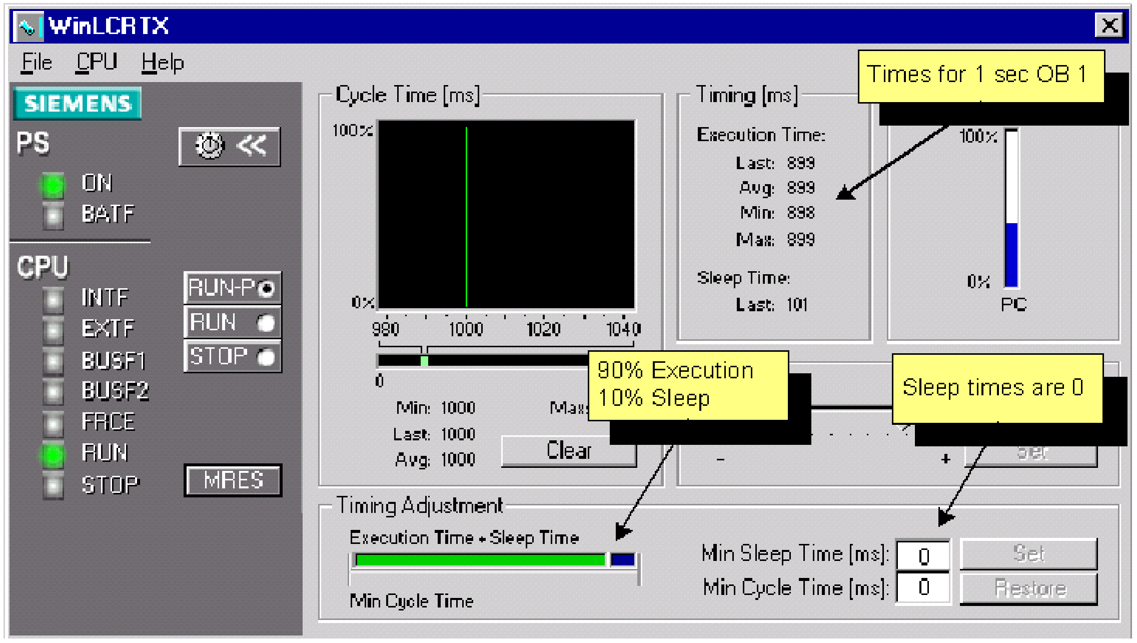
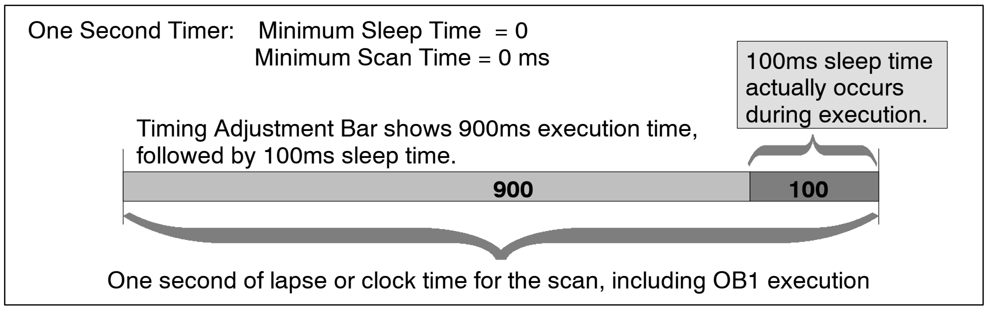


Figure D-7 One Second OB1 with no Sleep Time

The tuning bar shown below shows the amount of execution time and sleep time that occurs during the one second that OB1 executes. The tuning bar shows all the sleep time in one block for simplicity. In reality, as explained previously, the sleep time occurs in 1-millisecond intervals throughout the 1 second that it takes OB1 to complete.

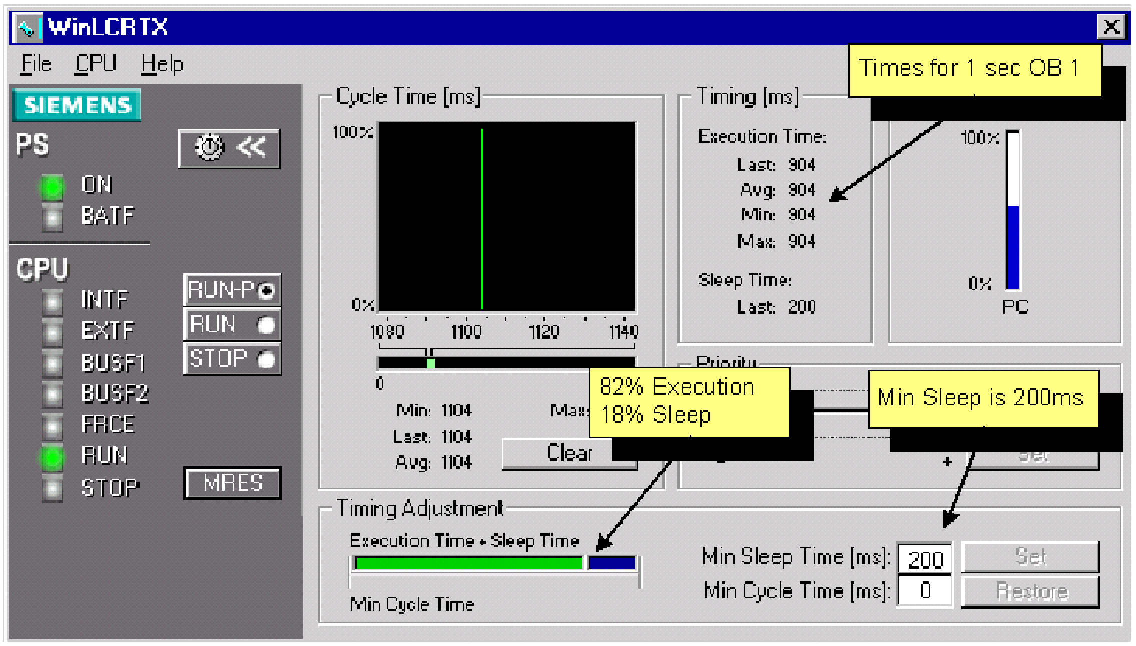

One Second Timer: Minimum Sleep Time = 0
Minimum Scan Time = 0 ms
100ms sleep time actually occurs during execution.
Timing Adjustment Bar shows 900ms execution time, followed by 100ms sleep time.
900
100
One second of lapse or clock time for the scan, including OB1 execution

Now consider this same example with the parameters set as shown in the following table.

Table D-2 WinLC RTX Parameters

WinLC RTX Configuration	Value
Number of CPUs	1
Tuning Panel Priority	50 (default WinLC RTX)
Tuning Panel Min Sleep Time	200 milliseconds
Tuning Panel Min Scan Time	0 milliseconds (default)
Sleep-Monitoring Wake Interval	9 milliseconds (default)
Sleep-Monitoring Sleep Interval	1 millisecond (default)

In this case, 100 milliseconds of sleep time occurs during the 1 second that OB1 takes to complete – the same as in the previous example. Because the minimum scan time parameter is 200, another 100 milliseconds of free cycle wait (sleep) time occurs after OB1 completes. The total scan time is now 1100 milliseconds, 1000 milliseconds (1 second) for OB1 to complete and another 100 milliseconds of free cycle wait time.



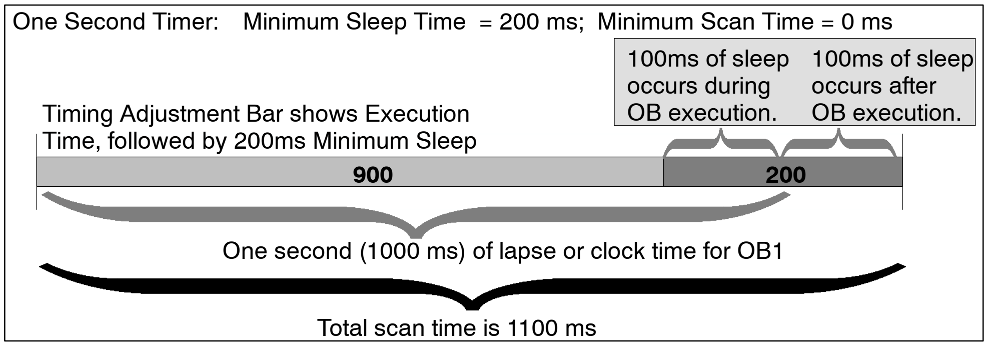
Min: 1104
Max:
Last: 1104
Avg: 1104

Execution Time:
Last: 904
Avg: 904
Min: 904
Max: 904
Sleep Time:
Last: 200

Min Sleep Time [ms]: 200
Min Cycle Time [ms]: 0

Figure D-8 One Second OB1 with 200ms Sleep Time

The following figure explains the timing adjustment bar in more detail for this case.


One Second Timer: Minimum Sleep Time = 200 ms; Minimum Scan Time = 0 ms
100ms of sleep occurs during OB execution.
100ms of sleep occurs after OB execution.
Timing Adjustment Bar shows Execution Time, followed by 200ms Minimum Sleep
900
200
One second (1000 ms) of lapse or clock time for OB1
Total scan time is 1100 ms

Example – Sleep-Monitoring Delaying OB Start Time

Consider a control program that checks a 20-millisecond timer in a loop in OB1 (or has other control logic) so that OB1 takes 20 milliseconds of clock time to complete. This program also has an OB35 that is configured to execute every 100 milliseconds. OB35 takes about 1 millisecond to execute.

Table D-3 WinLC RTX Parameters

WinLC RTX Configuration	Value
Number of CPUs	1
Tuning Panel Priority	50 (default WinLC RTX)
Tuning Panel Min Sleep Time	10 milliseconds (default)
Tuning Panel Min Scan Time	0 milliseconds (default)
Sleep-Monitoring Wake Interval	9 milliseconds (default)
Sleep-Monitoring Sleep Interval	1 millisecond (default)

This is a well-behaved application that has all the default settings for the tuning parameters. Figure D-9 shows what you would see in the tuning panel.

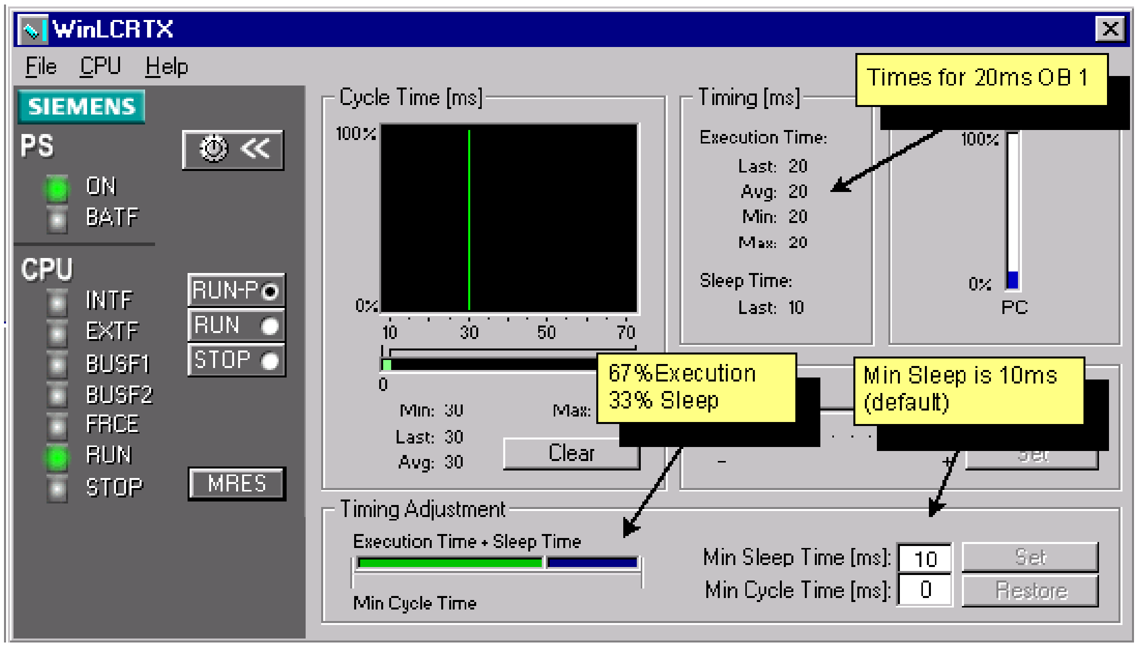


Figure D-9 20ms OB1 with 10ms Sleep Time

The tuning panel shows exactly what you would expect to see in this case. However, the tuning panel only shows the free cycle (OB1 execution sequence) information. You cannot see what is happening to OB35.

OB1 takes more than the 9-millisecond wake interval time to complete. In fact, the sleep monitor algorithm suspends the control program twice during each free cycle scan. When the sleep monitor suspends OB1, no OB can execute, including OB35. If the start time for OB35 occurs while OB1 is suspended, then OB35 must wait until the end of the suspension period – a maximum of 1 millisecond (sleep interval value). Similarly, OB35 could be suspended for 1 millisecond if the end of the execution monitor wake interval occurs while OB35 is executing. Figure D-10 illustrates this OB 35 “jitter.”

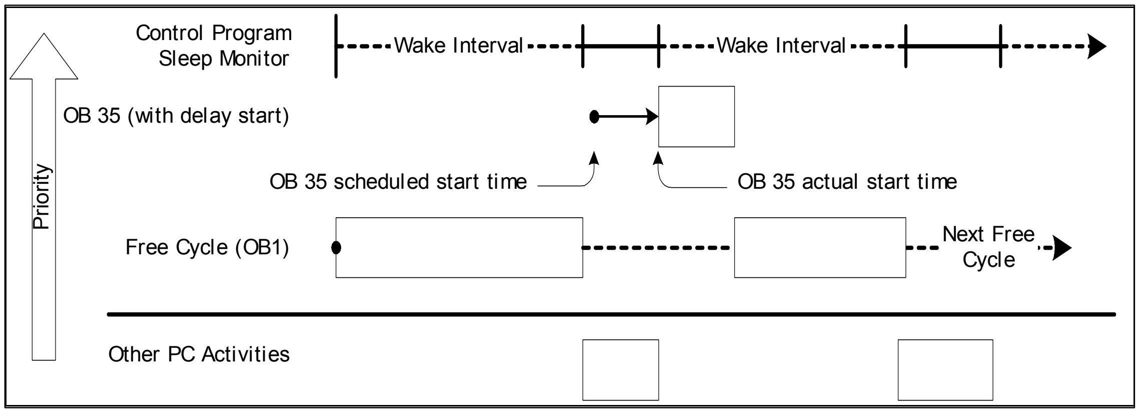


Figure D-10 Example with OB35 Delayed Start or ”Jitter”

For many applications, this 1 millisecond jitter in OB35 is acceptable. If the default WinLC RTX execution sleep monitor is a problem for your application, you can resolve it using the techniques described in the next section, Modifying Control Program Sleep Monitor Parameters or the last section, User Control Program Sleep Management.

Modifying Control Program Sleep Monitor Parameters

Many applications function well with the default values for the Sleep Monitor Parameters. On the other hand, if you have an application that has a scan cycle less than 50 milliseconds with sufficient free cycle sleep time (as described in section 5.1, Free Cycle Sleep Management), you may want to reduce or eliminate the effects of the sleep monitor described in the previous sections. You can do this by changing the sleep monitor parameters with the WinLC RTX Advanced Tuning application included with WinLC RTX releases beginning with WinLC RTX 3.1.

Note

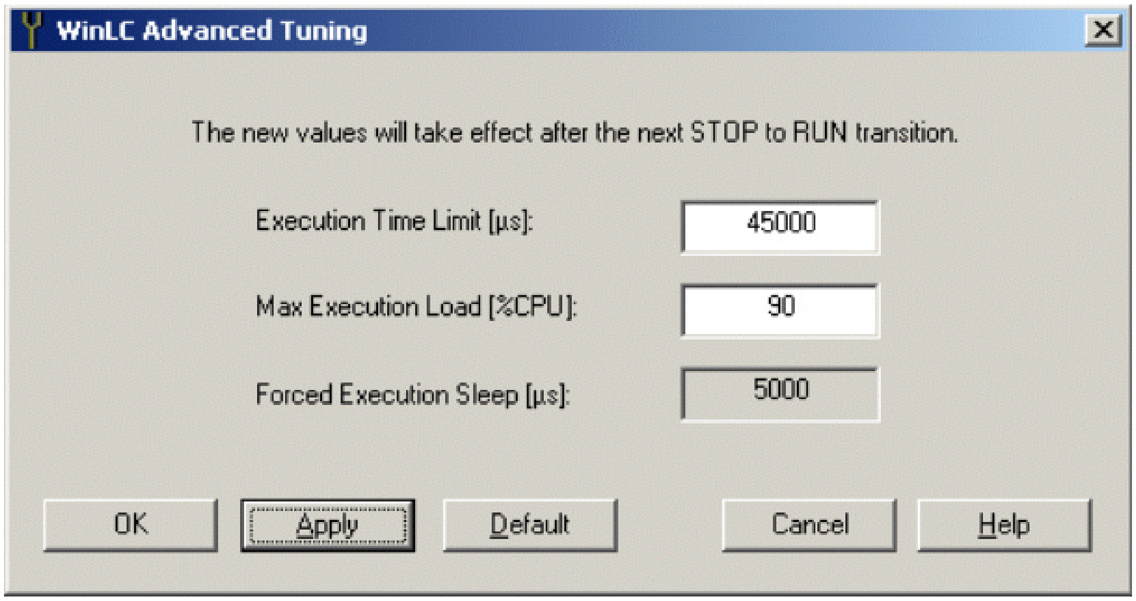
The path for the application WinLC RTX Advanced Tuning is C:\Siemens\WinAC\WinLCRTX\AdvancedTuning.exe for most installations of WinLC RTX.

WinLC RTX must be running before you start the application. The Advanced Tuning application does not need an argument if you are using WinLC RTX.

If your application is like the one described in the example Sleep-Monitoring Delaying OB Start Time, you can use the Advanced Tuning application to eliminate the effects of the sleep monitor.

This sample application requires 20 milliseconds to execute OB 1 and has 10 milliseconds Min Sleep Time, for a total free cycle time of 30 milliseconds. OB 35 and other interrupt OBs make the total scan time more than this, depending on how fast the interrupt OBs execute. In this example, you can assume that the longest total scan time would be less than 45 milliseconds. So in this case, you would enter 45000 microseconds as the Execution Time Limit in the Advanced Tuning application. You do not want the control program to ever use more than 90% of the PC CPU time, so leave the Max Execution Load at the default 90% value.

Figure D-11 shows the Advanced Tuning application with these values.


WinLC Advanced Tuning

The new values will take effect after the next STOP to RUN transition.

Execution Time Limit [µs]: 45000

Max Execution Load [%CPU]: 90

Forced Execution Sleep [µs]: 5000

Figure D-11 WinLC RTX Advanced Tuning Application

After you apply these parameters, the control program monitor does not affect this sample program. As shown in Figure D-12, the sample program meets the requirement of using less than 90% of the CPU during every 45-millisecond execution time limit (or wake interval).

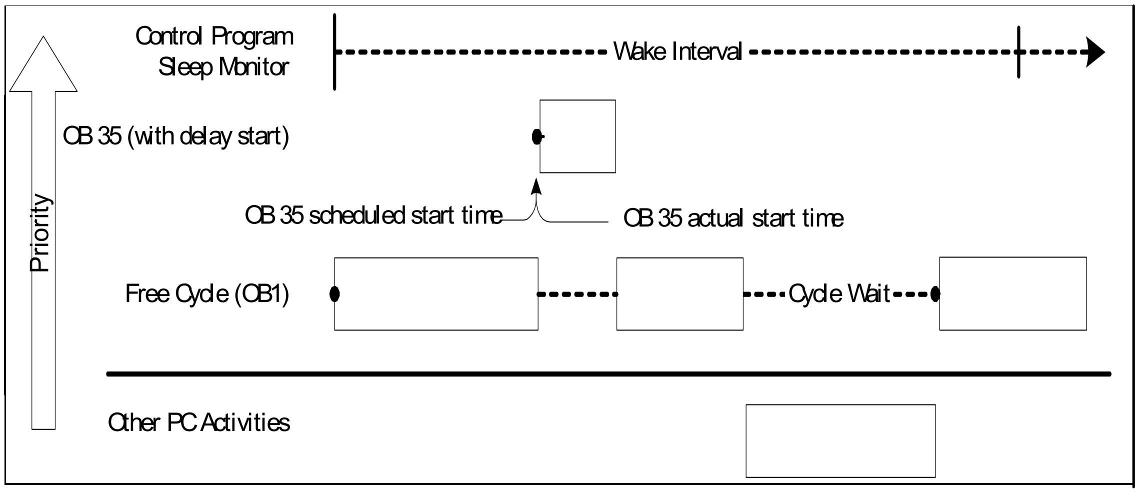


Figure D-12 Example Eliminating OB35 ”Jitter” Using Modified Sleep Monitor Parameters

User Control Program Sleep Management

All S7 CPUs, including WinLC RTX, support another execution sleep management mechanism using SFC 47 WAIT. In concept, this mechanism can be used to implement a solution similar to the control program sleep monitor described in previous sections. This solution provides greater control over which OBs will be affected by the wait (or sleep) time. This solution requires that you insert the appropriate sleep times into your program.

To understand how to use SFC47 to insert sleep times, reconsider the example Sleep-Monitoring Delaying OB Start Time.

In this example, the control program has control logic in OB1 that takes 20 milliseconds to execute. This program also has an OB35 that is configured to execute every 100 milliseconds. OB35 takes about 1 millisecond to execute. The WinLC RTX parameters are set according to the table below.

Table D-4 WinLC RTX Parameters

WinLC RTX Configuration	Value
Number of CPUs	1
Tuning Panel Priority	50 (default WinLC RTX)
Tuning Panel Min Sleep Time	10 milliseconds (default)
Tuning Panel Min Scan Time	0 milliseconds (default)
Sleep-Monitoring Wake Interval	9 milliseconds (default)
Sleep-Monitoring Sleep Interval	1 millisecond (default)

With this configuration, a 1-millisecond jitter occurs in the OB35 start time and execution time because the OB1 execution time (20 ms) is greater than the monitor wake interval (9 ms). When the control program execution monitor detects that no sleep time has occurred in the wake interval, it forces the control program to sleep for the monitor sleep interval time (1 ms). Figure D-10 above illustrates this jitter effect on OB35.

You can change this example application so that sleep intervals are inserted into the free cycle without affecting the start time of interrupt OBs such as OB35. You do this by implementing a periodic sleep interval (like the control program sleep monitor) at an OB priority greater than the free cycle, but lower priority than the interrupt OB (implementing a cyclic OB with a call to SFC 47 WAIT).

The steps below describe how to change this example so that the free cycle (OB1) does not use more than 50% of the CPU and the cyclic OB35 starts immediately, even if the start time is in a sleep interval.

1. Add OB36 to the control program. Assuming OB35 executes in 1 ms, a 2 ms wait allows OB35 to execute during the wait and still leaves 1 ms of wait time. This is enough to satisfy the control program execution monitor requirement for sleep time. OB36 contains the following simple program.

   ```
   CALL “WAIT”  // SFC 47 wait function
    WT: 3000    // 3000 microseconds or 3 milliseconds
   ```

2. Change the configuration of OB36 in the STEP 7 Hardware Configuration Editor. In the WinLC RTX Properties dialog/Cyclic Interrupt tab, set the OB36 priority to a number lower than OB35 (2) and set the Execution (ms) to 6.
3. Set the minimum sleep time in the tuning panel to zero unless you need the free cycle CPU load to be less than 50%.

Figure D-13 illustrates the benefit of this small application change.

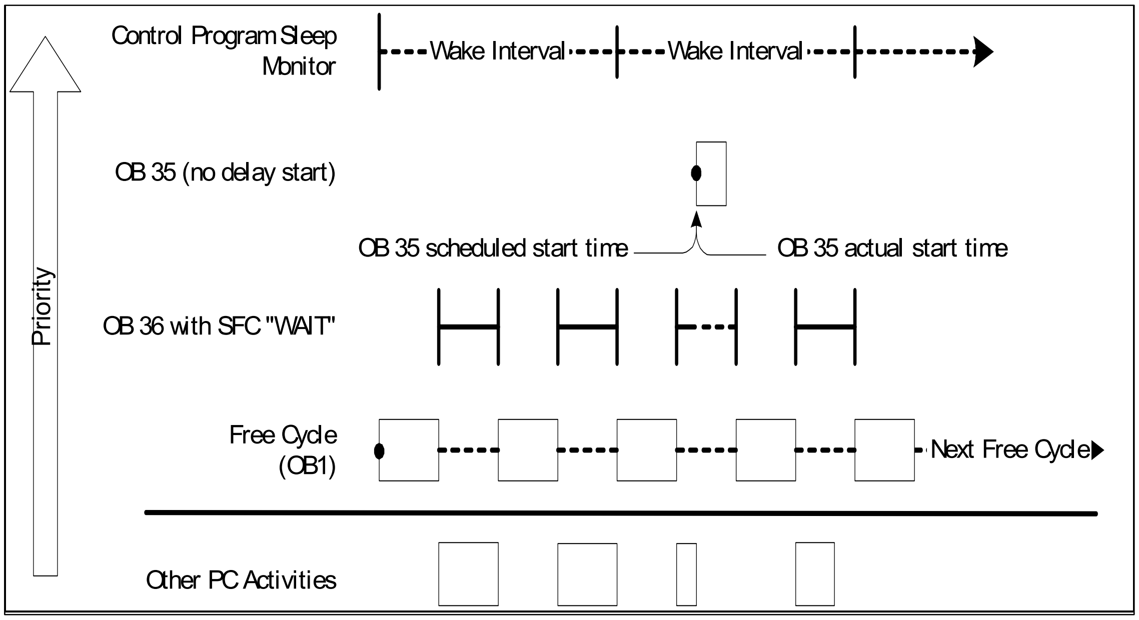


Figure D-13 Example Eliminating OB35 ”Jitter” Using OB36

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: