

SIMATIC

Computing

User Manual

This manual is part of the documentation package with order no:
6ES7 673-6CC01-8BA0

Getting Started with Computing	1
Product Overview	2
Setting Up Computing Software	3
Using Computing to Access Data	4
Accessing the Process Data with the Data Control	5
User Controls	6
S7 Diagnostics Buffer Control (DBuffer)	7
Designing Simple Process Forms with the WinAC SoftContainer	8
Creating Tag Files with the TagFile Configurator	9
Memory Areas of the S7 Controllers	A
Properties and Methods	B
Events	C
Using the Computing Configuration Tool	D
Using Computing with DCOM	E
Guidelines for Programming with Computing	F
Using Control Engine Strings	G

A5E00065508-04

Edition: 06/2000

Safety Guidelines

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



Danger

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



Warning

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



Caution

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

Note

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

Qualified Personnel

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

Correct Usage

Note the following:



Warning

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

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

Trademarks

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

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

Copyright Siemens AG 1999–2000 All rights reserved

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

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

Siemens Aktiengesellschaft

Disclaimer of Liability

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

© Siemens AG 1999–2000
Technical data subject to change.

A5E00065508

Preface

The Computing software uses the ActiveX (also known as OLE) technology of Microsoft to provide you with access to the data provided by your control engine. The Computing software consists of the following:

- A set of SIMATIC controls, which are ActiveX or OCX (OLE custom controls) controls for accessing control engines
- An OPC (OLE for process control) server that allows other OPC applications to access the data stored in the control engine (such as WinLC of WinAC Basis or the CPU 416-2 DP ISA of WinAC Pro)
- TagFile Configurator for creating tag files that allow symbolic addressing and remote access to multiple control engines
- A configuration tool for configuring remote access
- An OLE container (SoftContainer) for creating process forms with the SIMATIC controls

Note

As used by the Computing software, the term “control engine” applies to a processor or program that manages and manipulates data which is used to control a process or machine. The control engine can be either software or hardware.

WinAC Basis provides a Windows Logic Controller (WinLC) as its control engine, and WinAC Pro provides a slot PLC as its control engine. (The term “slot PLC” in the manual refers to a slot PLC such as CPU 416–2 DP ISA or CPU 416–2 DP ISA Lite. In the manual, the CPU416–2 DP ISA Lite is treated identically to the CPU416–2 DP ISA.) The ActiveX controls provided by Computing communicate with these control engines, as well as other SIMATIC S7 controllers.

Audience

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

Scope of the Manual

This manual describes the features and the operation of version 3.0 of the Computing software.

How to Use This Manual

This manual provides information focused for different audiences. Not only are there two methods for accessing the process data (either using the ActiveX controls or the OPC interface), but there are also different levels of complexity for each method. You can either use the control provided, or you can write programs that include the controls.

If you will be using the ActiveX (OCX) controls in a container application such as Visual Basic, refer to Getting Started (Chapter 1) and Product Overview (Chapter 2).

The chapters for the specific SIMATIC controls provide information about Configuring the controls. Appendix B describes the properties and methods for the controls, and Appendix C describes the events.

If you will be using the OPC interface:

- If you will be connecting an existing (third-party) OPC client application to the WinAC products, refer to the product overview (Chapter 2) for the name of the OPC server object.
- If you will be designing a client application for use with the WinAC products, refer to the OPC specification (*OLE for Process Control Data Access Standard, version 2.0* from the OPC Foundation).

Other Manuals

You can find information in the online help for the Computing software. For additional 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 about the structure of the operating system and how to design a user program for WinLC. Use this manual when creating a user program with the STEP 7 automation software.
OPC Server Interface Manual	This manual describes the browsable OPC server interface provided with the Computing software.
Windows Logic Controller (WinLC) User Manual	This manual provides basic information about the performance characteristics and operation of the WinLC controller.
WinAC Controlling with CPU 416–2 DP ISA Hardware and Installation Manual	This manual provides basic information about the performance characteristics and operation of the CPU 416–2 DP ISA controller.

Additional Assistance

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

To contact Customer Service for Siemens in North America:

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

To contact Customer Service for Siemens in Europe:

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

Contents

1	Getting Started with Computing	1-1
1.1	Overview	1-2
1.2	Creating a Sample I/O Panel	1-4
1.3	Connecting Third-Party Controls to a Data Control	1-12
1.4	Using Computing with Microsoft Excel	1-15
1.5	Using the SoftContainer Provided by Computing	1-19
2	mpiProduct Overview	2-1
2.1	Product Overview	2-2
2.2	Using an ActiveX Control to Access the Process Data	2-4
2.3	Connecting to Your Process with the OPC Server	2-6
3	Setting Up Computing Software	3-1
3.1	Overview	3-2
3.2	Authorization	3-3
3.3	Installing and Uninstalling the Computing Software	3-5
3.4	Connecting Computing to a Slot PLC or Communications Card	3-7
4	Using Computing to Access Data	4-1
4.1	Accessing Data in Control Engines	4-2
4.2	Accessing a Local Control Engine	4-4
4.3	Accessing a Remote Control Engine	4-5
4.4	Communicating with Multiple Control Engines	4-6
5	Accessing the Process Data with the Data Control	5-1
5.1	Connecting the SIMATIC Controls to the Control Engine	5-2
5.2	Configuring the Connection Properties for the Data Control	5-3
5.3	Selecting the Control Engine for the Data Control	5-4
5.4	Connecting the ActiveX Controls to the Control Engine	5-9
5.5	Filtering the Properties for the ActiveX Controls	5-13
5.6	Configuring Custom Events	5-15
5.7	Creating a Connection Table	5-16
5.8	Sample Program for Creating a Connection Table and an Event Table	5-17
5.9	Sample Program for Responding to Events	5-19
5.10	Sample Programs for Reading and Writing Data	5-23
5.11	Sample Program for Reading and Writing Boolean Data	5-28
5.12	Properties, Methods, and Events of the Data Control	5-29

6	User Controls	6-1
6.1	Connecting the User Controls to the Process Data	6-2
6.2	Using the Property Pages of the Button Control	6-4
6.3	Properties and Methods of the Button Control	6-9
6.4	Events of the Button Control	6-10
6.5	Using the Property Pages of the Edit Control	6-11
6.6	Properties and Methods of the Edit Control	6-18
6.7	Events of the Edit Control	6-19
6.8	Error Codes for the Edit Control	6-20
6.9	Using the Property Pages of the Label Control	6-21
6.10	Properties and Methods of the Label Control	6-26
6.11	Events of the Label Control	6-26
6.12	Using the Property Pages of the Slider Control	6-27
6.13	Properties and Methods of the Slider Control	6-34
6.14	Events of the Slider Control	6-35
7	S7 Diagnostic Buffer Control (DBuffer)	7-1
7.1	Accessing the S7 Diagnostics Buffer	7-2
7.2	Configuring the DBuffer Control	7-4
7.3	Properties and Methods of the DBuffer Control	7-7
8	Designing Simple Process Forms with the SoftContainer	8-1
8.1	Starting the SIMATIC Computing SoftContainer	8-2
8.2	Creating a Process Form	8-4
8.3	Switching from Design Mode to Run Mode	8-6
8.4	Saving Your Process Form	8-8
9	Creating Tag Files with the TagFile Configurator	9-1
9.1	Connecting to Multiple Control Engines over DCOM	9-2
9.2	Using Symbols to Access Data in the Control Engine	9-5
9.3	Creating a Tag File	9-6
9.4	Configuring a Tag File for Local or Remote Access to a Control Engine .	9-10
9.5	Changing the Control Engine Symbol Name in the Tag File Editor	9-13
A	Memory Areas of the S7 Controllers	A-1
A.1	Memory Areas of S7 Controllers	A-2
A.2	Accessing the S7 Data Types	A-3
A.3	Descriptions of the S7 Data Types	A-6

B	Properties and Methods in work	B-1
B.1	AboutBox Method	B-1
B.2	Activated Property	B-1
B.3	Alignment Property	B-2
B.4	Appearance Property	B-3
B.5	AutoConnect Property	B-3
B.6	AutoConnectTimeout Property	B-4
B.7	BackColor Property	B-5
B.8	bDiagBuffOK Property	B-5
B.9	bEngineConnected Property	B-6
B.10	BorderStyle Property	B-6
B.11	Caption Property	B-7
B.12	Connect Method	B-7
B.13	ConnectName Method	B-8
B.14	ConnectObject Method	B-9
B.15	ControlEngine Property	B-10
B.16	DataFormat Property	B-11
B.17	DefaultDeadband Property	B-12
B.18	DefaultUpdateRate Property	B-13
B.19	Direction Property	B-14
B.20	Disconnect Method	B-14
B.21	DisplayFormatButtons Property	B-15
B.22	DisplayHelpButton Property	B-15
B.23	DisplayHelpOnEventButton Property	B-16
B.24	DisplayLowerPanel Property	B-17
B.25	DisplayUpdateButton Property	B-17
B.26	DisplayUpperPanel Property	B-18
B.27	DisplayValue Property	B-18
B.28	Enabled Property	B-19
B.29	EnableSort Property	B-19
B.30	Factor Property	B-20
B.31	FalseCaption Property	B-21
B.32	FalseColor Property	B-21
B.33	FalsePicture Property	B-22
B.34	Font Property	B-22

B.35	ForeColor Property	B-23
B.36	FormatDisplay Property	B-23
B.37	KnobHeight Property	B-24
B.38	KnobPicture Property	B-24
B.39	KnobWidth Property	B-25
B.40	LargeChange Property	B-25
B.41	Locked Property	B-26
B.42	Max and Min Properties	B-26
B.43	MultipleEngines Property	B-27
B.44	Offset Property	B-27
B.45	PCName Property	B-28
B.46	Picture Property	B-28
B.47	PopUpHelp Method	B-29
B.48	PopUpHelpOnEvent Method	B-29
B.49	Precision Property	B-30
B.50	PropertyChangedName Method	B-30
B.51	PropertyChangedObject Method	B-31
B.52	PushButton Property	B-32
B.53	RawMax and RawMin Properties	B-32
B.54	ReadMultiVariables Method	B-33
B.55	ReadVariable Method	B-33
B.56	ScaleMode Property	B-34
B.57	SelectEvent Method	B-35
B.58	ShowErrorBoxes Property	B-36
B.59	ShowMinMax Property	B-36
B.60	SmallChange Property	B-37
B.61	StretchMode Property	B-38
B.62	Style Property	B-39
B.63	TagSource Property	B-39
B.64	Text Property	B-40
B.65	Ticks Property	B-40
B.66	TrueCaption Property	B-40
B.67	TrueColor Property	B-41
B.68	TruePicture Property	B-42
B.69	Update Method	B-42

B.70	Value Property	B-43
B.71	WriteMode Property	B-43
B.72	WriteNow Method	B-44
B.73	WriteMultiVariables Method	B-44
B.74	WriteVariable Method	B-45
B.75	Zeropad Property	B-45
C	Events	C-1
C.1	Change Event	C-1
C.2	Click Event	C-1
C.3	ConnectionError Event	C-1
C.4	DbClick Event	C-2
C.5	Error Event	C-2
C.6	KeyDown Event	C-3
C.7	KeyPress Event	C-4
C.8	KeyUp Event	C-5
C.9	MouseDown Event	C-6
C.10	MouseMove Event	C-7
C.11	MouseUp Event	C-8
C.12	ValueChanged Event	C-9
D	Using the Computing Configuration Tool	D-1
D.1	Configuring the OPC Connection	D-2
D.2	Selecting the Language	D-5
D.3	Selecting the Control Engine for Older Programs	D-6
D.4	Setting up Communications Using the PG/PC Interface	D-7
E	Using Computing with DCOM	E-1
E.1	Using DCOM to Provide Remote Access	E-2
E.2	Configuring the Permissions for the Server Computer	E-4
E.3	Configuring the Permissions for the Client Computer	E-14
E.4	Troubleshooting	E-20

F	Guidelines for Programming with Computing	F-1
F.1	Guidelines for Third-Party Containers	F-2
F.2	Programming Guidelines	F-4
F.3	Guidelines for Creating Custom ActiveX Controls	F-6
F.4	Using a Custom ActiveX Control with a Data Control	F-7
F.5	Known Problems for Computing Version 3	F-10
G	Using Control Engine Strings	G-1

Index

Figures

1-1	Using Computing to Access Data in the Control Engine	1-2
1-2	Sample Program ("Counters") for the Application Examples	1-3
1-3	Adding SIMATIC Controls to the VB Toolbox	1-5
1-4	Sample I/O Panel Created in Visual Basic	1-6
1-5	Displaying the Expanded List of Properties	1-7
1-6	Using the Filter Button	1-7
1-7	Adding Properties to the Filter	1-8
1-8	Applying the Filter to the Property List	1-8
1-9	Assigning a Variable in the Control Engine to a Property in a Control ...	1-9
1-10	Connecting the Data Control to a Control Engine	1-10
1-11	Operating the Sample I/O Panel	1-11
1-12	Adding the Data Control to the Visual Basic Toolbox	1-12
1-13	Assigning a Variable to the Caption Property of a VB Label Control	1-13
1-14	Connecting to the Control Engine (Label Control Example)	1-14
1-15	Adding an Event Key to the Data Control	1-17
1-16	Assigning a Variable to an Event Key	1-17
1-17	Inserting a SIMATIC Control into the SoftContainer	1-19
1-18	Using the SoftContainer to Create a Sample I/O Panel	1-20
1-19	Connecting to the Control Engine (SoftContainer Example)	1-21
1-20	Assigning the Value Property to a Variable	1-22
1-21	Configuring the Display Properties of the Edit control	1-23
1-22	Placing the Container into Run Mode	1-24
2-1	Accessing the Process Data with Computing	2-2
2-2	Applications Working with Many OPC Servers	2-6
2-3	Using the OPC Server to Access Your Process Data	2-7
3-1	Setting the PG/PC Interface for Slot PLC CPU 416-2 DP ISA (local) ...	3-7
4-1	Accessing Data in Control Engines	4-3
4-2	Accessing a Local Control Engine	4-4
4-3	Accessing a Remote Control Engine	4-5
4-4	Accessing Data in Several Remote Control Engines	4-7
5-1	Using the Data Control for Connecting to a Control Engine	5-2
5-2	Data Control Properties (General Tab)	5-3
5-3	Data Control Properties (Engine Tab)	5-4
5-4	Selecting a Tag File for the Data Control	5-6
5-5	Direct Connection for a Local or a Remote Computer	5-7
5-6	Configuring DCOM for a Specific Control Engine	5-8
5-7	Entering a Symbol for the Assigned Variable	5-10
5-8	Browsing to a Symbol in the Tag File	5-11
5-9	Data Control Properties (Connections Tab)	5-13
5-10	Data Control Properties (Connections Tab)	5-14
5-11	Data Control Properties (Connections Tab)	5-14
5-12	Data Control Properties (Events Tab)	5-15
5-13	Sample Program for Responding to Events in the Control Engine	5-19
6-1	Assigning Variables for a Button, Edit, Control	6-2
6-2	Assigning Variables for the Edit Control	6-3
6-3	Button Control Properties (General Tab)	6-5
6-4	Button Control Properties (Picture Tab)	6-5
6-5	Button Control Properties (Font Tab)	6-6
6-6	Button Control Properties (Color Tab)	6-7
6-7	Button Control Properties (Name Tab)	6-8
6-8	Edit Control Properties (General Tab)	6-12

Figures, continued

6-9	Edit Control Properties (Scaling Tab)	6-14
6-10	Edit Control Properties (Font Tab)	6-15
6-11	Edit Control Properties (Color Tab)	6-16
6-12	Edit Control Properties (Name Tab)	6-17
6-13	Label Control Properties (General Tab)	6-22
6-14	Label Control Properties (Picture Tab)	6-23
6-15	Label Control Properties (Font Tab)	6-24
6-16	Label Control Properties (Color Tab)	6-25
6-17	Label Control Properties (Name Tab)	6-25
6-18	Slider Control Properties (General Tab)	6-28
6-19	Orientation of the Slider Control	6-28
6-20	Elements of the Slider Control	6-30
6-21	Slider Control Properties (Scaling Tab)	6-30
6-22	Slider Control Properties (Picture Tab)	6-31
6-23	Slider Control Properties (Color Tab)	6-32
6-24	Slider Control Properties (Name Tab)	6-33
7-1	Accessing the diagnostics Buffer of an S7 Controller	7-2
7-2	Elements of the DBuffer Control	7-3
7-3	"Diagnostics Buffer" Tab of the DBuffer Control	7-4
7-4	"Visibility" Tab of the DBuffer Control	7-5
7-5	DBuffer Control Properties (Name Tab)	7-6
8-1	Container with the Default Process Form	8-2
8-2	Elements of the Status Bar	8-3
8-3	Inserting a Control from the Toolbar	8-4
8-4	Inserting a Third-Party Control into the Process Form	8-5
8-5	Changing the Container to Run Mode	8-7
8-6	Saving a Process Form for Computing	8-8
9-1	Connecting to Multiple Control Engines over DCOM	9-2
9-2	Creating a Tag File for Multiple Control Engines	9-3
9-3	Configuring the Data Control for Multiple Control Engines	9-4
9-4	Using Symbols to Access Data in the Control Engine	9-5
9-5	TagFile Configurator	9-6
9-6	Inserting a SIMATIC Program into the Tag File	9-7
9-7	Inserting a New Control Engine into the Tag File	9-8
9-8	Configuring a Control Engine for Local Access	9-11
9-9	Configuring a Control Engine for Remote Access	9-12
9-10	Configuring a Control Engine for Remote Access	9-13
A-1	Accessing Data by Byte, Word, and Double Word	A-3
A-2	Accessing Data in an ARRAY	A-7
A-3	Accessing the DATE Data Type	A-7
A-4	Accessing the DATE_AND_TIME Data Type	A-8
A-5	Accessing STRING Data	A-9
A-6	Accessing the TIME Data Type	A-10
A-7	Accessing the TIME_OF_DAY (TOD) Data Type	A-10
D-1	Configuring the OPC Connection	D-3
D-2	Selecting the Language for the CPU Panel and Help Files	D-5
D-3	Selecting the Control Engine	D-6
D-4	Accessing the PG/PC Interface	D-7
D-5	Setting the PG/PC Interface for PC Internal (local)	D-8
D-6	Setting the PG/PC Interface From the STEP 7 Computer	D-9
D-7	Setting the PG/PC Interface From WinLC	D-11

Figures, continued

E-1	Using SIMATIC Computing on a Single Computer	E-2
E-2	Using Computing over DCOM	E-3
E-3	Tasks for Configuring the DCOM Server	E-4
E-4	Distributed COM Configuration Properties	E-5
E-5	Configuring the Default Access Permissions for DCOM	E-6
E-6	Changing the Access Permissions for Users or Groups	E-7
E-7	Configuring the Default Launch Permissions for DCOM	E-8
E-8	Changing the Launch Permissions for Users or Groups	E-9
E-9	Selecting the Running Class for DCOM	E-10
E-10	Configuring the DCOM Access Permissions for the Server	E-11
E-11	Changing the Access Permissions for Users or Groups	E-12
E-12	Configuring the DCOM Identity Permissions for the Server	E-13
E-13	Tasks for Configuring the DCOM Client	E-14
E-14	Distributed COM Configuration Properties	E-15
E-15	Configuring the Default Access Permissions for DCOM	E-16
E-16	Changing the Access Permissions for Users or Groups	E-17
E-17	Configuring the Default Launch Permissions for DCOM	E-18
E-18	Changing the Launch Permissions for Users or Groups	E-19
F-1	Connecting to the Control Engine (Scrollbar Control Example)	F-9

Tables

1-1	Assigning Sample Addresses to the SIMATIC Controls	1-9
1-2	Assigning Sample Addresses to the SIMATIC Controls	1-21
2-1	Standard Controls Provided by Computing	2-4
5-1	Sample Program for Manually Creating a Connection	5-17
5-2	Sample Program for Manually Creating an Event	5-18
5-3	Sample Program for Creating a Connection for Responding to Events ..	5-20
5-4	Sample Program for Responding to Events in the Control Engine	5-21
5-5	Other Subroutines for Running the Sample Program	5-22
5-6	Reading a Single Variable from the Control Engine	5-24
5-7	Writing a Single Variable to the Control Engine	5-24
5-8	Sample Program for Reading an Array of Variables	5-25
5-9	Sample Program for Writing an Array of Variables	5-26
5-10	Reading Multiple Variables in the Control Engine	5-27
5-11	Writing Multiple Variables to the Control Engine	5-27
5-12	Reading and Writing Multiple Variables	5-28
5-13	Properties and Methods of the Data Control	5-29
5-14	Events of the Data Control	5-30
5-15	Data Control Error Codes	5-30
6-1	Properties and Methods of the Button Control	6-9
6-2	Events of the Button Control	6-10
6-3	Size of Data Types for the Edit control	6-13
6-4	Properties and Methods of the Edit Control	6-18
6-5	Events of the Edit Control	6-19
6-6	Error Codes for the Edit Control	6-20
6-7	Properties and Methods of the Label Control	6-26
6-8	Events of the Edit Control	6-26
6-9	Properties and Methods of the Slider Control	6-34
6-10	Events of the Slider Control	6-35
7-1	Properties and Methods of the DBuffer Control	7-7
A-1	Memory Areas of the S7 controllers	A-2
A-2	Addressing the S7 Data Types and S7 Memory Areas	A-4
A-3	S7 Data Types as C or Visual Basic Data Types	A-5
B-1	Settings for the Data Format	B-11
C-1	SCodes (Error Event Codes)	C-3
D-1	OPC Error Codes	D-4
F-1	Reading and Writing a Changed Value of a Property	F-6
F-2	Sample Program for an ActiveX Control Used with Computing	F-8
F-3	Error Codes	F-12
G-1	Identifying the Type of Control Engine	G-1

Getting Started with Computing

Chapter Overview

The Computing software provides you with a variety of ways to access and to use data from a control engineer, such as an S7 CPU, the Windows Logic Controller (WinLC) of WinAC Basis, or a slot PLC such as the CPU 416-2 DP ISA of WinAC Pro.

This chapter provides some easy programming examples to help you become familiar with the power and flexibility that can be achieved by using the ActiveX controls provided by SIMATIC Computing. You can find the sample programs in the following directory on the drive where you installed the Computing software:

`[C:] \Siemens \WinAC \Examples`



Warning

After you assign a variable to the Value property of a SIMATIC or a third-party ActiveX control, the control is able to access process data. When you change the value that is displayed in the control, you are changing the value in the actual process. Do not connect this example to a control engine that is connected to equipment.

Altering process data can cause unpredictable process operation, and unpredictable process operation could result in death or serious injury to personnel, and/or damage to equipment.

Exercise caution to ensure that you do not access any data that could cause process equipment to operate erratically. Always install a physical emergency stop circuit for your machine or process.

Section	Description	Page
1.1	Overview	1-2
1.2	Creating a Sample I/O Panel	1-4
1.3	Connecting Third-Party Controls to a Data Control	1-12
1.4	Using Computing with Microsoft Excel	1-15
1.5	Using the SoftContainer Provided by Computing	1-19

1.1 Overview

Computing allows you not only to access the data in the control engine, but also allows you flexibility in how you access the data and what you can do with the data.

The examples in this chapter show some of the ways you can use the ActiveX controls provided by Computing. As shown in Figure 1-1, this chapter provides samples of subroutines for the following applications:

- Create a user interface: You can use a the SIMATIC control with a third-party container (such as Microsoft's Visual Basic) to create an I/O interface panel. See Section 1.2. (You can use this panel to test the other sample programs in this chapter.)
- Use a standard ActiveX control: You can also use a standard control (such as a Label control from Visual Basic) to access data in the control engine. See Section 1.3.
- Load data from the control engine into standard software packages: You can load data into a Microsoft Office application (such as Microsoft's Excel). See Section 1.4.

Instead of using the third-party container (Section 1.2), you can use the SoftContainer provided by Computing to create a simple I/O panel. See Section 1.5.

You can find the sample programs in the following directory on the drive where you installed the Computing software: **[C:] \Siemens \WinAC \Examples**

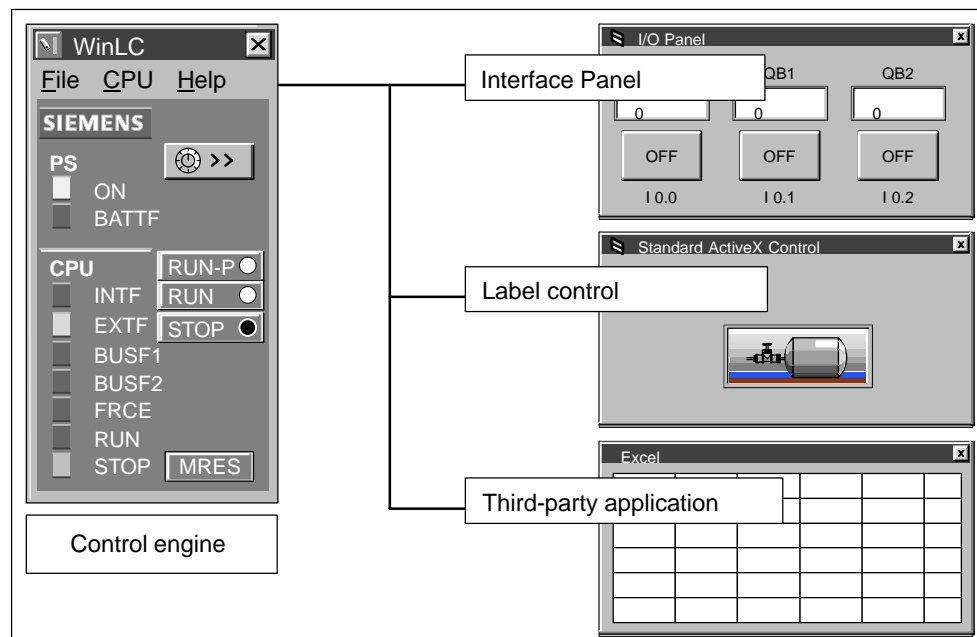


Figure 1-1 Using Computing to Access Data in the Control Engine

Sample Program Used with the Application Examples

Figure 1-2 shows the sample program used by the application examples. The program uses the following logic:

- If Input bit 0.0 (I 0.0) is on, the program increments a value stored in MB1 and moves the new value to QB0.
- If Input bit 0.1 (I 0.1) is on, the program decrements a value stored in MB3 and moves the new value to QB1.
- If Input bit 0.2 (I 0.2) is on, the program increments a value stored in MB5 and moves the new value to QB2.

Use STEP 7 to create and download this program to the control engine.

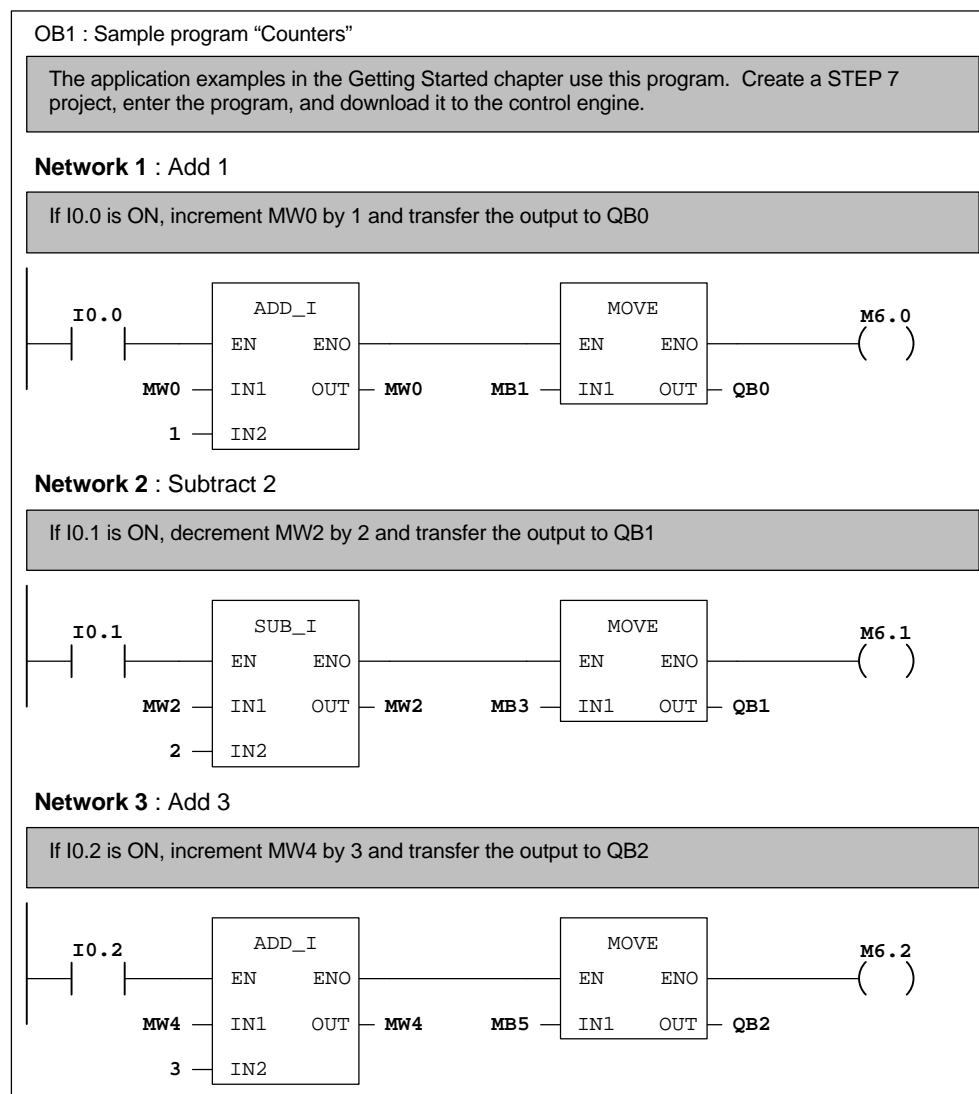


Figure 1-2 Sample Program ("Counters") for the Application Examples

1.2 Creating a Sample I/O Panel

The Data control allows any ActiveX container (such as Visual Basic 5.0) to access data in the control engine. You can use the SIMATIC controls provided by Computing with Visual Basic to create a simple I/O panel that interacts with a program running on a control engine

To create this sample application, you need the following items:

- Microsoft Visual Basic 5 or higher
- SIMATIC controls from Computing
- Control engine: for example, WinLC or a slot PLC such as CPU 416-2 DP ISA
- Sample program (see Section 1.1)
- STEP 7 (to download the program to the control engine)



Caution

Using the timer function improperly or using breakpoints in Visual Basic with Computing can cause problems that could potentially cause your computer or application to crash or lock up. Depending on the configuration, this could cause the application to lose communication with the control engine. This could cause unpredictable process operation, which could result in death or serious injury to personnel, and/or damage to equipment.

Always install a physical emergency stop circuit for your machine or process.

Concerning VB timers: The Timer function in Visual Basic version 5 allows a timer to interrupt code in progress within the same thread, which can cause problems with potentially serious consequences. If you use VB timers with Computing, observe the following guidelines:

- Always kill (disable) the timers in the Form_Unload subroutine. Otherwise, a timer can trigger an event while the VB program is shutting down; this condition could cause your computer or your application to crash, lock up, or to continue running invisibly.
 - If you start your timer in the Form_Load subroutine, the timer event could occur before the other objects have been instantiated. In order to ensure that the objects have been properly instantiated, always start a timer in the Form_Load subroutine with a large interval (such as 1 or 2 seconds) to allow the objects to be properly instantiated. Subsequent timer intervals can be set to shorter intervals.
-

Inserting the SIMATIC Controls into the Toolbox for Visual Basic

Use the following procedure to create the sample I/O panel:

1. Open a standard Visual Basic project:
 - Select the **File ► New Project** menu command to display the “New Project” dialog box.
 - Select the “Standard EXE” icon and click on the “Open” button.
2. Select the **Project ► Components** menu command to display the “Components” dialog box.
3. As shown in Figure 1-3, select the following SIMATIC controls in the “Components” dialog box:
 - Data control (Siemens SIMATIC Data Control)
 - Panel control (Siemens S7 Panel Control, which comes with WinLC or a slot PLC)
 - Diagnostic Buffer (Siemens SIMATIC Diagnostic Buffer Control)
 - User controls (Siemens SIMATIC UserControls) The icons for Button, Label, Slider, and Edit appear in the Icon tab.
4. Click on the “Apply” button. The SIMATIC controls that you selected appear in the toolbox for Visual Basic. Click on the “OK” button to close the “Components” dialog box.

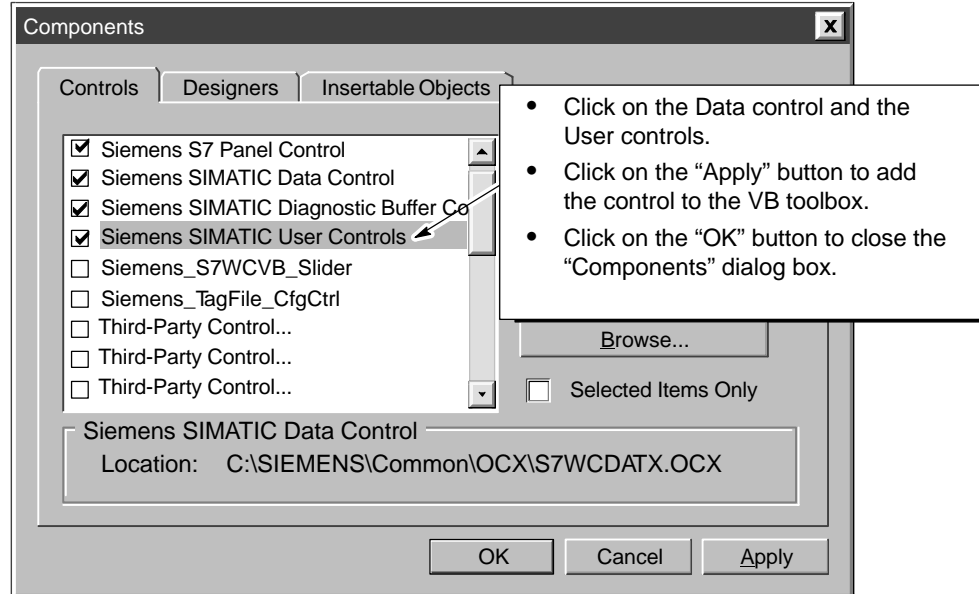


Figure 1-3 Adding SIMATIC Controls to the VB Toolbox

Creating the VB Form for the Sample I/O Panel

1. Insert one Data control, three Edit controls and three Button controls onto the Visual Basic form. See Figure 1-4.
2. Create standard VB label controls to indicate the address that you have assigned for each of the controls. See Figure 1-4.

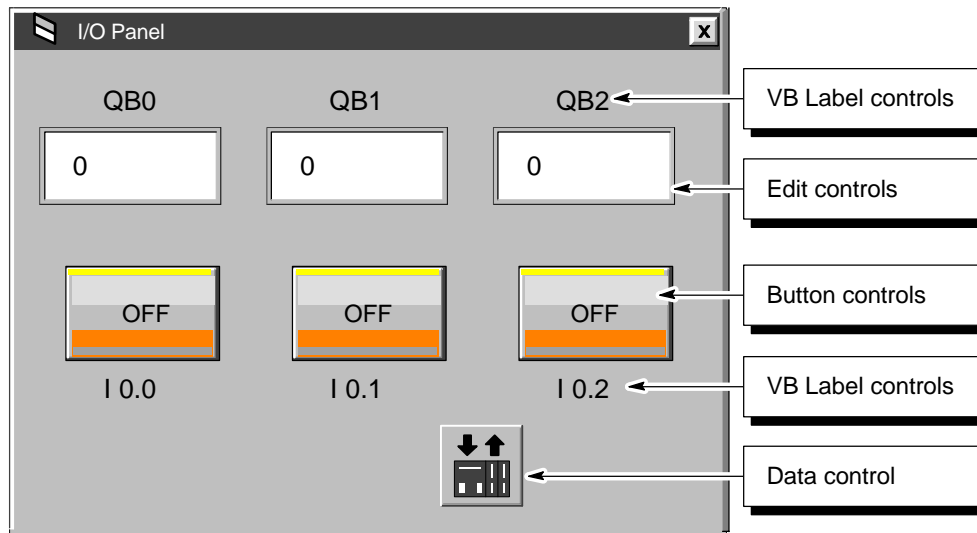


Figure 1-4 Sample I/O Panel Created in Visual Basic

Assigning Variables in the Control Engine to the SIMATIC Controls

In order to connect the SIMATIC or third-party controls to the process data in the control engine, you must assign a variable (memory location in the control engine) to the Value property (or to other properties) for each control. You use the Connection tab of “Properties” dialog box for the Data control to assign variables in the control engine. You cannot assign a variable to the Value property of a control by using the property list of the control itself.

Use the following procedure to assign variables to the SIMATIC controls:

1. Select the Data control and click the right mouse button to bring up the pop-up menu. From the pop-up menu, select the **Properties** menu command to display the “Properties” dialog box for the Data control.
2. Select the “Connections” tab. Click on the “+” symbol to expand the list of controls.
3. As shown in Figure 1-5, select the control and click on its “+” symbol to expand its properties list.

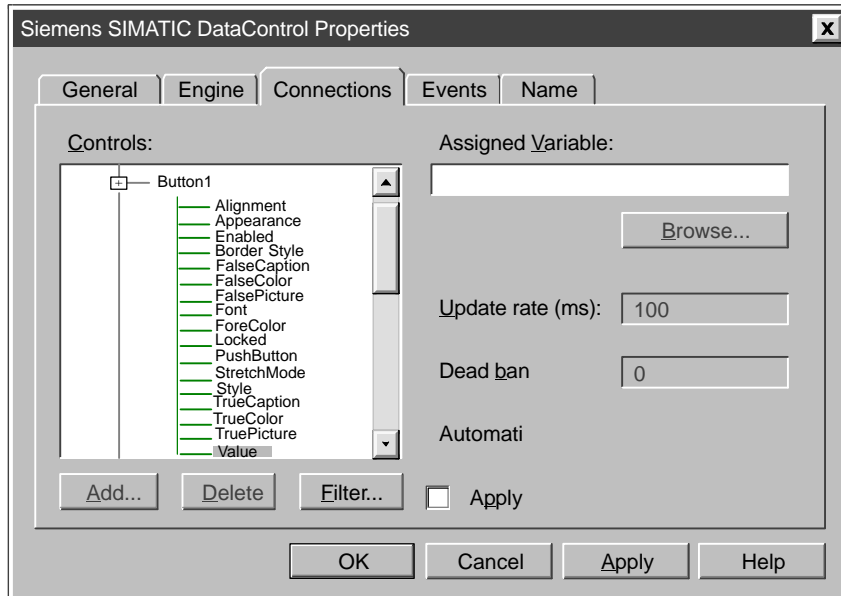


Figure 1-5 Displaying the Expanded List of Properties

4. As shown in Figure 1-6, click on the "Filter" button.

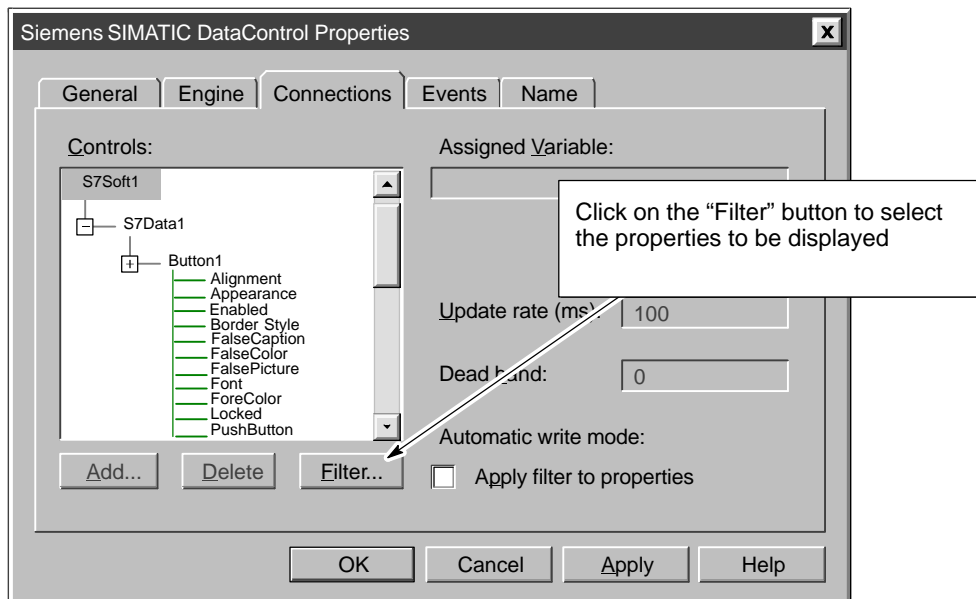


Figure 1-6 Using the Filter Button

5. As shown in Figure 1-7, enter the properties to display and click on the "Add" button. Use the "Edit" button to correct entries and the "Delete" button to remove entries.

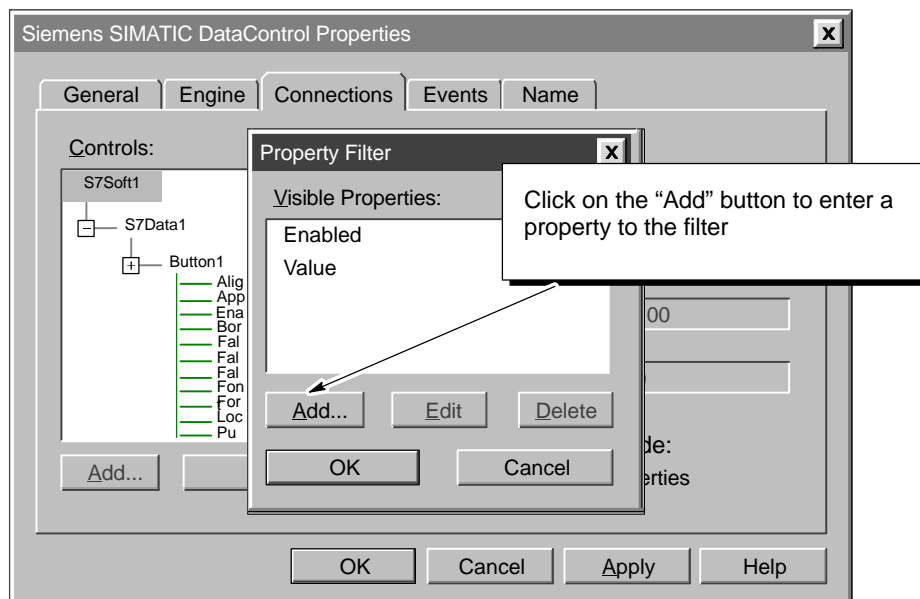


Figure 1-7 Adding Properties to the Filter

6. As shown in Figure 1-8, select the “Apply filter to properties” check box to display only the properties listed in the filter. You can use the “Apply filter to properties” check box to toggle the filter on and off.

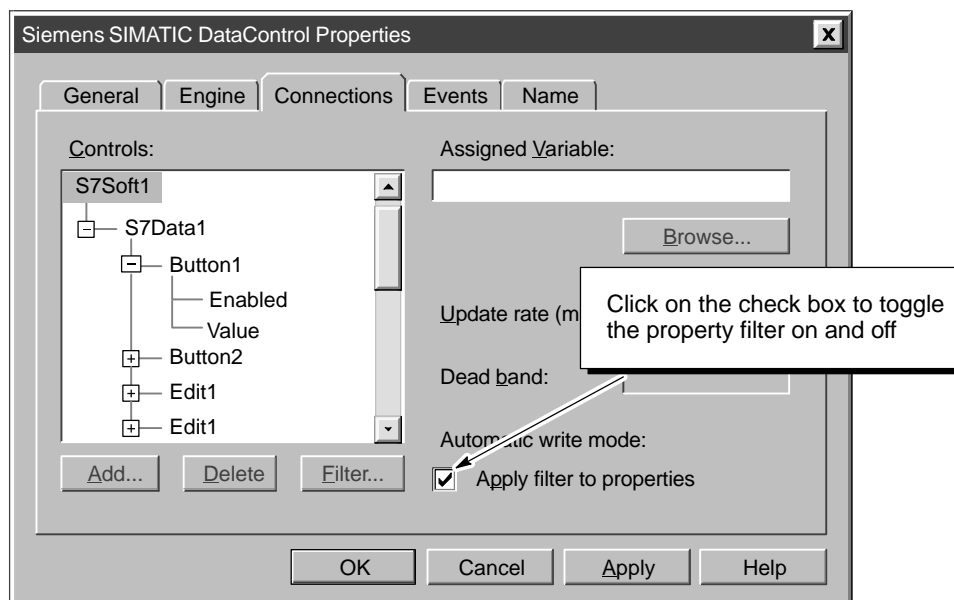


Figure 1-8 Applying the Filter to the Property List

7. As shown in See Figure 1-9, select the Value property of the control.

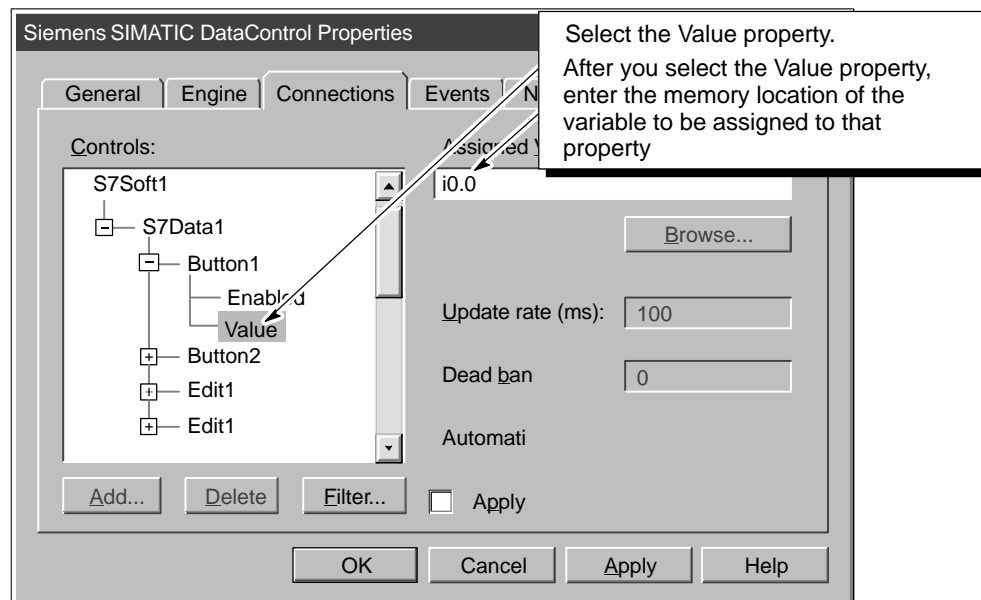


Figure 1-9 Assigning a Variable in the Control Engine to a Property in a Control

8. Refer to Table 1-1 and assign the variables (memory addresses in the control engine) to the SIMATIC controls.
9. Click on the “Apply” button to enter the assigned variables.

Table 1-1 Assigning Sample Addresses to the SIMATIC Controls

Control	Address	Description
Edit1	QB0	Output value of first counter
Edit2	QB1	Output value of second counter
Edit3	QB2	Output value of third counter
Button1	I 0.0	Enable bit for first counter
Button2	I 0.1	Enable bit for second counter
Button3	I 0.2	Enable bit for third counter

Selecting a Control Engine

Use the following procedure to configure the Data control for communicating with the control engine:

1. Select the “Engine” tab to configure the control engine. See Figure 1-10.
2. Select the “Direct Connect” option and enter either the control engine, for example, `winLC` or `wcS7=3` (for a slot PLC such as the CPU 416-2 DP ISA). Click on the “Apply” button to enter the data, and then click on the “OK” button to close the dialog box.

Note

`wcS7=3` is identical to `S7DosIntf/MPI=3`, which is still enabled for compatibility.

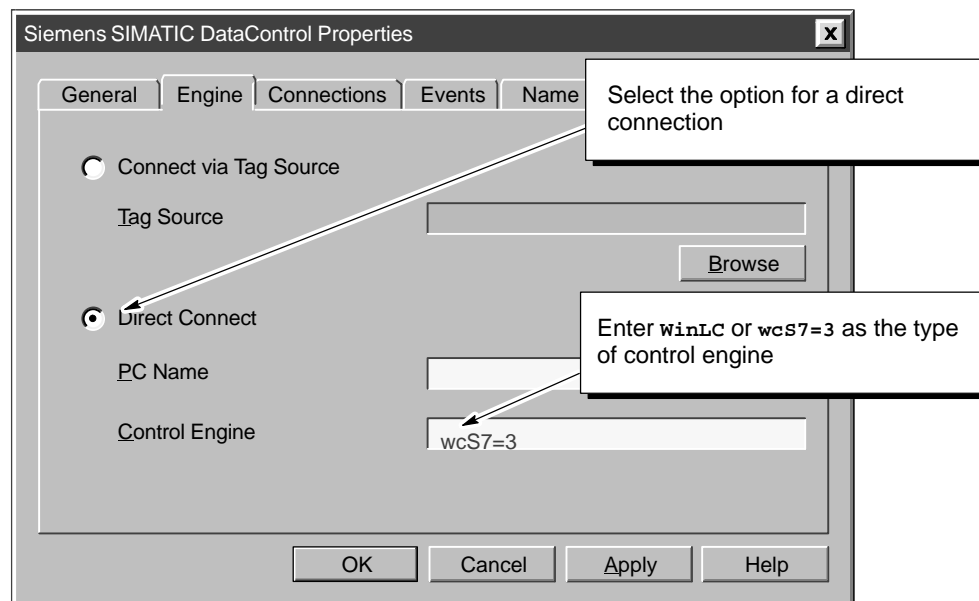


Figure 1-10 Connecting the Data Control to a Control Engine

Operating of the I/O Panel Program

Before you run the I/O Panel program, make certain that the control engine is running the sample program “Counters”.

Note

If the control engine (for example, WinLC or a slot PLC such as the CPU 416-2 DP ISA) is not active, the Data control cannot make a connection. Before setting Visual Basic into Run Mode, ensure that the control engine is running.

1. Select the **File ► Save Project** menu command to save the program before switching Visual Basic from Design mode to Run mode.
2. Click on the “Start” icon or select the **Run ► Start** menu command to switch Visual Basic from Design mode to Run mode to run the I/O panel program.
3. Click on the Button Control for I 0.0 to start the first counter. See Figure 1-11.
 - The Button control changes color to show the state of I 0.0.
 - The Edit Control for QB0 displays the counter value.
4. Click on the Button control for I 0.1 to start the second counter. See Figure 1-11.
 - The Button control changes color to show the state of PI 0.1.
 - The Edit control for QB1 displays the counter value.
5. Click on the Button control for I 0.2 to start the third counter. See Figure 1-11.
 - The Button control changes color to show the state of I 0.2.
 - The Edit control for QB2 displays the counter value.

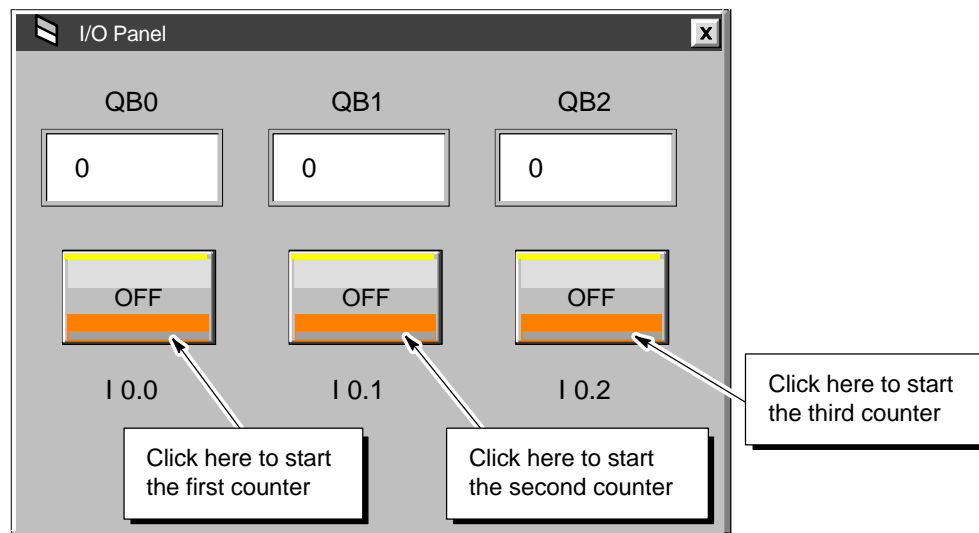


Figure 1-11 Operating the Sample I/O Panel

1.3 Connecting Third-Party Controls to a Data Control

You can use the Data control to connect any ActiveX control (such as a VB Label control) to data in the control engine. To create this sample application, you need the following items:

- Microsoft Visual Basic 5 or higher
- Data control from Computing
- Control engine: for example, WinLC or a slot PLC such as CPU 416-2 DP ISA
- Sample program (see Section 1.1)
- STEP 7 (to download the program to the control engine and to turn on the peripheral input bits of the sample program)

You can also use the sample I/O Panel application to turn on the peripheral input bits of the sample program running in the control engine. See Section 1.2 for information about the I/O Panel application.

Creating a VB label that Displays a Value in the Control Engine

Use the following procedure to connect a Data control with a Label control:

1. Open a standard Visual Basic project: Use the **File ► New Project** menu command to display the “New Project” dialog box, then select the “Standard EXE” icon and click on the “Open” button.
2. Add the Data control to the VB toolbox. For information about adding controls to the VB toolbox, see Section 1.1 and Figure 1-12.

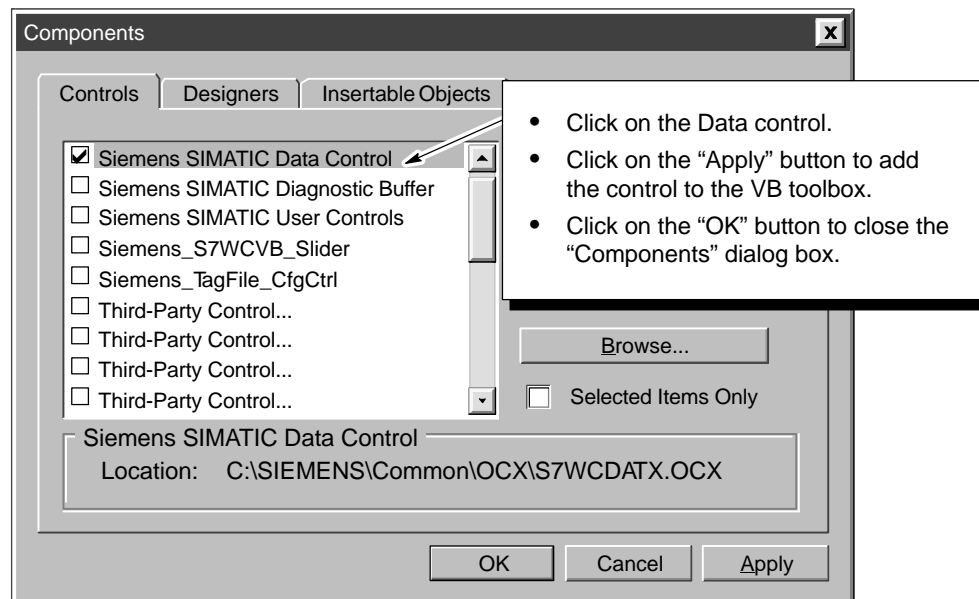


Figure 1-12 Adding the Data Control to the Visual Basic Toolbox

3. Insert a Data control onto the VB form. (For information about inserting controls onto the VB form, see Section 1.1.)
4. Insert a VB label on your form. Change the Border Style property to "1-Fixed Single."
5. Select the Data control and click the right mouse button to bring up the pop-up menu. From the pop-up menu, select **Properties** to display the "Properties" dialog box for the Data control.
6. From the "Properties" dialog box, select the "Connections" tab. Click on the "+" symbol to expand the list of controls.
7. Select the Label1 control and click on its "+" symbol to expand its properties list.
8. Select the Caption property and enter **QB0** in the "Assigned Variable" field. See Figure 1-13. Click on the "Apply" and "OK" buttons to enter the data and close the "Properties" dialog box.

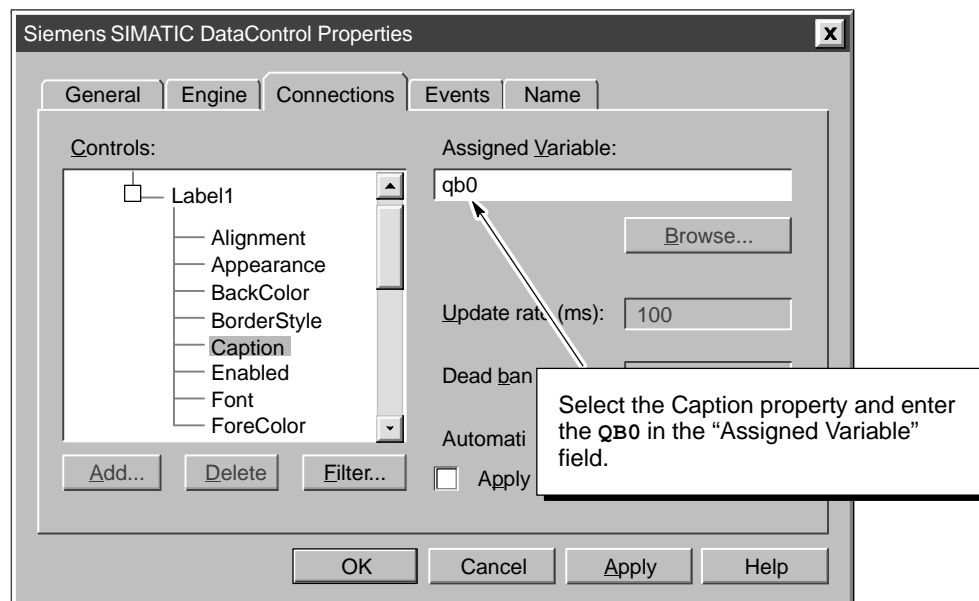


Figure 1-13 Assigning a Variable to the Caption Property of a VB Label Control

Running the Sample Program for the Label Control

Save the program before switching Visual Basic from Design mode to Run mode. When the sample program runs, the caption of the label displays the value of QB0 in the control engine.

Note

If the control engine (for example, WinLC or a slot PLC such as the CPU 416-2 DP ISA) is not active, the Data control cannot make a connection. Before setting Visual Basic into Run Mode, ensure that the control engine is running.

Use the following procedure to configure the Data control for communicating with the control engine and for running the sample program.

1. Select the "Engine" tab to configure the control engine. See Figure 1-14.
2. Select the "Direct Connect" option and enter either **winLC** or **wcs7=3** (for a slot PLC such as the CPU 416-2 DP ISA) for the control engine. Click on the "Apply" button to enter the data, and then click on the "OK" button to close the dialog box.
3. Switch Visual Basic from Design mode to Run mode to run the sample program.

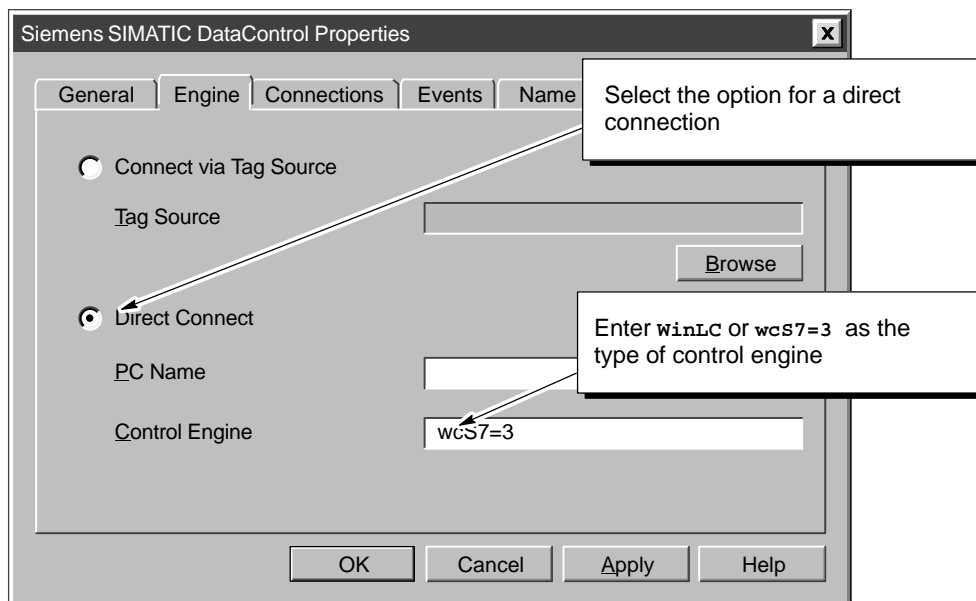


Figure 1-14 Connecting to the Control Engine (Label Control Example)

1.4 Using Computing with Microsoft Excel

Using the Data control in an Excel spreadsheet allows you to access the values in the control engine. To create this sample application, you need the following items:

- Microsoft Excel 97 or Excel 2000
- Control engine: either WinLC, an S7 CPU in Excel, or a slot PLC such as CPU 416-2 DP ISA
- Sample program (see Section 1.1)
- STEP 7 (to download the program to the control engine and to turn on the peripheral input bits of the sample program)

This example shows how to use events to call code to update your Excel cells. Events are a means of tying changing data to code within a VBA form.

Note

You can also use the I/O Panel application to turn on the peripheral input bits of the sample program running in the control engine. See Section 1.2 for information about the I/O Panel application.

Creating a Command Button in Excel

The first step in creating the sample Excel application is to create a command button. Use the following procedure to create a command button:

1. Start the Excel application. (If prompted about whether to enable or disable macros, select the “Enable Macros” option.)
2. In the following cells of the spreadsheet, enter the following labels:
 - In cell A1, enter: **qb0**
 - In cell A2, enter: **qb1**
 - In cell A3, enter: **qb2**
3. Select the **View ► Toolbars ► Control Toolbox** menu command to display the Control toolbox.
4. Click on the “Design Mode” icon in the Control toolbox to put the spreadsheet into Design mode.
5. Insert a Command Button control onto the spreadsheet by clicking on the “Command Button” icon in the Control toolbox and then clicking the left mouse button in an empty area of the spreadsheet.
6. Move or size the Command Button control as required.

Using the Visual Basic Editor to Configure the Command Button

After you have created the command button, you use the Visual Basic Editor in Excel to configure the command button for starting and stopping the program.

Use the following procedure to configure the command button:

1. Select the command button (CommandButton1).
2. Select the **Tools ► Macro ► Visual Basic Editor** menu command to display the Visual Basic editor.
3. In the Properties window, select the Caption property of CommandButton1 and enter the following caption:

Start Counting

4. Display the Code window by selecting the **View ► Code** menu command. Select "CommandButton1" from the drop-down list for the Object field. Enter the following code for the CommandButton1_Click() event:

UserForm1.show

5. Close the Code window for CommandButton1.

Creating a SIMATIC Data Control

1. Create a new user form by selecting the **Insert ► UserForm** menu command.
2. In the "Toolbox" window, click the right mouse button to bring up a pop-up menu and select the **Additional Controls...** menu command. (To display the "Toolbox" window, select the **View ► Toolbox** menu command.)
3. Scroll through the list of controls and select the Siemens Data control (by selecting the check box). Click on the "OK" button to insert the Data control onto the toolbox.
4. Select the "Data control" icon in the "Toolbox" window and insert a Data control onto UserForm1.
5. Select the Data control and click the right mouse button to bring up the pop-up menu. From the pop-up menu, select the **Properties** menu command to display the properties for the Data control (S7Data1) in the Properties window.
6. In the Properties window for S7Data1, select the "(Custom)" property field and then click on the expansion button to display the "Properties" dialog box for the Data control.

Adding Events for the Data Control

1. In the "Properties" dialog box for the Data control, select the "Events" tab. In the list under the "Keys" heading, select S7Data1.
2. Click on the "Add" button to add a new event key. See Figure 1-15. In the "Add" dialog box, enter QB0 in the "Add a new key" field.

After you click on the "OK" button, the event key is added to the S7Data1 control.

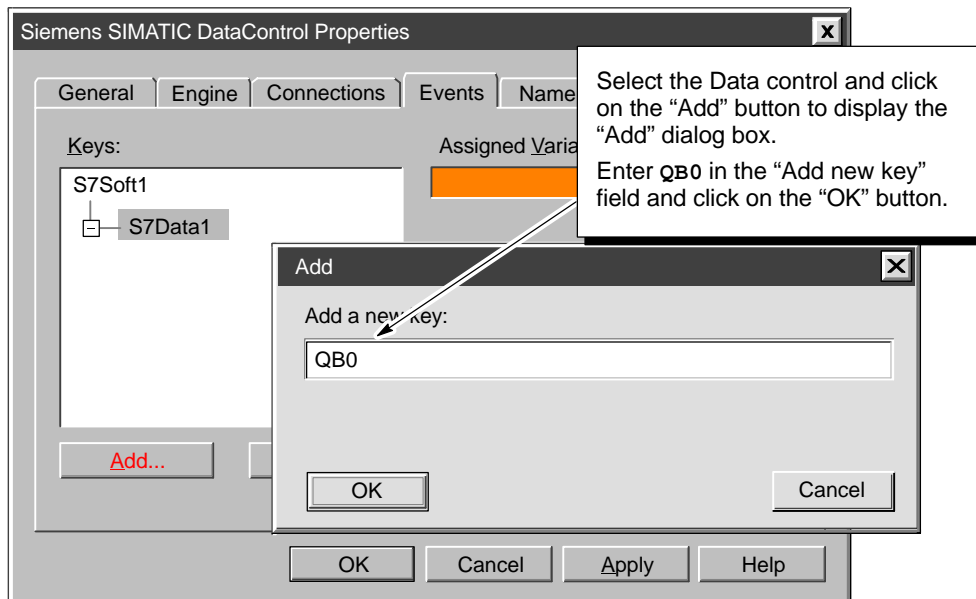


Figure 1-15 Adding an Event Key to the Data Control

3. In the "Properties" dialog box, enter memory address **QB0** in the "Assigned Variable" field. See Figure 1-16.
4. Click on the "Apply" button to accept the assigned variable. Notice that the event key "QB0" appears in boldface under S7Data1.
5. Enter new event keys for QB1 (memory address **QB1**) and QB2 (memory address **QB2**) by selecting S7Data1 again and repeating steps 2. and 3.

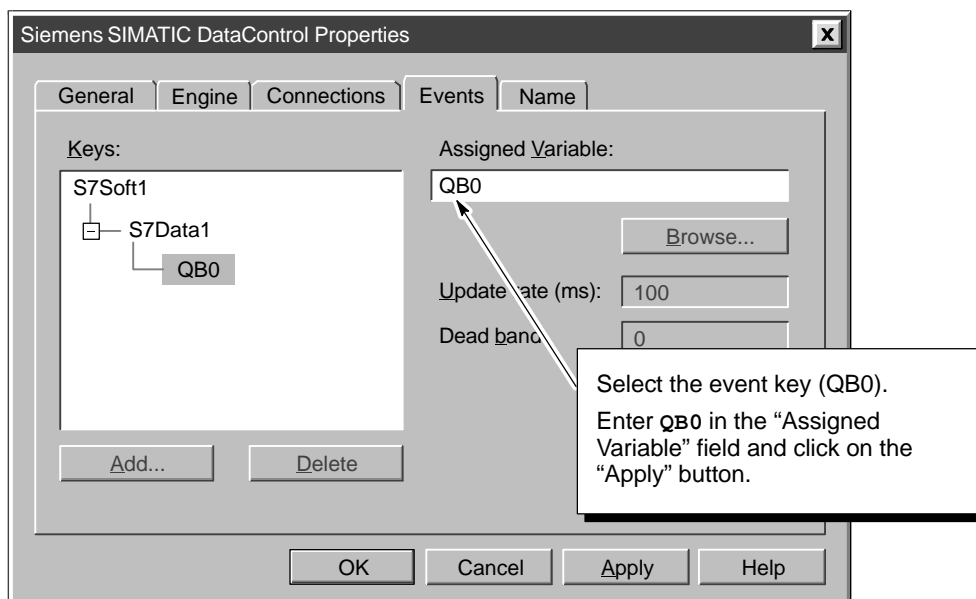


Figure 1-16 Assigning a Variable to an Event Key

Configuring the Control Engine for the Data Control

1. In the “Properties” dialog box for the Data control, select the “Engine” tab to configure the control engine.
2. Select the “Direct Connect” option and enter either `winLC`, an S7 CPU in Excel, or `wcs7=3` (for a slot PLC such as the CPU 416-2 DP ISA) as the control engine. Click on the “Apply” button to enter the data, and then click on the “OK” button to close the dialog box.

Entering a Sample Program for the Data Control

1. Select the Data control in UserForm1.
2. Select the **View ► Code** menu command to display the code window for the Data control.
3. Select S7Data1 from the drop-down list of the Object field.
4. For the S7Data1_ValueChanged event, enter the following program:

```
Select Case Property
    Case "QB0"
        Worksheets("Sheet1").Range("B1").Value = Value
    Case "QB1"
        Worksheets("Sheet1").Range("B2").Value = Value
    Case "QB2"
        Worksheets("Sheet1").Range("B3").Value = Value
End Select
```

5. Close the Code window for the Data control and close UserForm1.

Running the Sample Program

1. Select the **File ► Close and Return to Microsoft Excel** menu command to return to the spreadsheet.
2. Exit Design mode by clicking on the “Exit Design Mode” icon in the “Toolbox” window.
3. Connect the Excel spreadsheet to the control engine by clicking on the “Start Counting” command button.
4. Use the sample I/O panel (see Section 1.2) to start and stop the sample program in the control engine.

Note

To exit Excel or to activate the Excel menus, you must first close UserForm1.

1.5 Using the SoftContainer Provided by Computing

Computing provides a simple OLE container application (SoftContainer) for displaying and modifying the data from the control engine. Using this container, you can quickly insert the SIMATIC controls into a process form. (A process form is the SoftContainer document, or file, with the various controls.) No code can be written with this tool.

In order to run this sample process form, you need to have downloaded the sample program (see Section 1.1) to the control engine.

Inserting SIMATIC Controls into the Process Form

To start the Computing software, select the **Simatic ► PC Based Control ► Computing Softcontainer** menu command from the Start menu. The SoftContainer opens with the default process form (S7Soft1). You will insert the SIMATIC controls into this process form. See Figure 1-17.

1. In the toolbar, click on the icon for the Data control. (Moving the arrow pointer over an icon and then keeping it stationary for a second will display a tooltip for identifying the icon.)
2. Move the arrow pointer to the open process form. Notice that once the cursor moves inside the process form, the arrow pointer changes to a cross-hair pointer.
3. Click the left mouse button to insert the Data control.

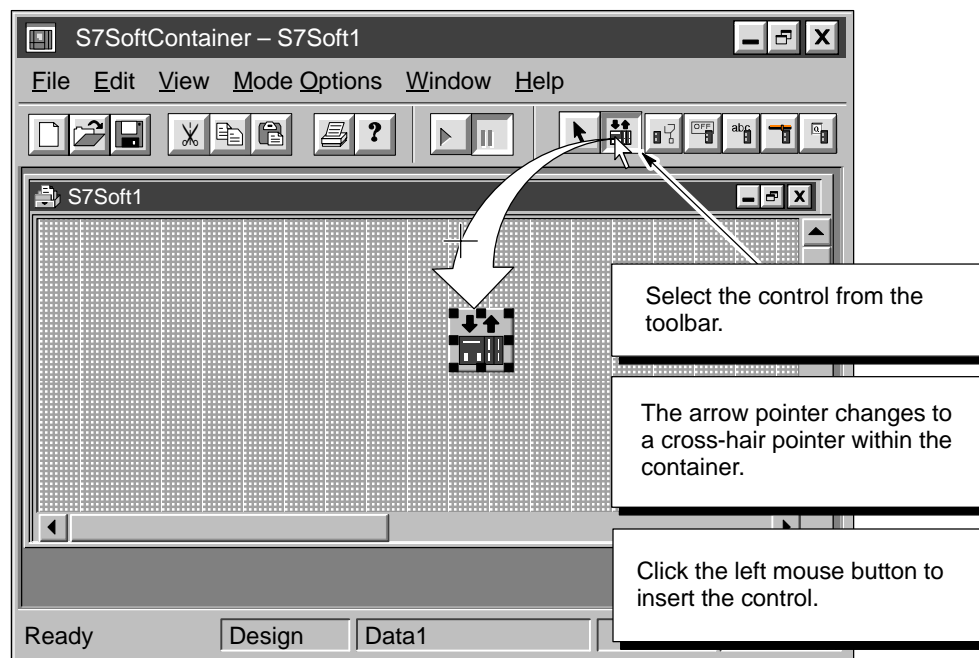


Figure 1-17 Inserting a SIMATIC Control into the SoftContainer

Repeat these steps to insert three Button controls and three Edit controls. (For more information about inserting controls into the SoftContainer, see Section 8.2.) Figure 1-18 shows a sample layout for the controls in the process form (S7Soft1).

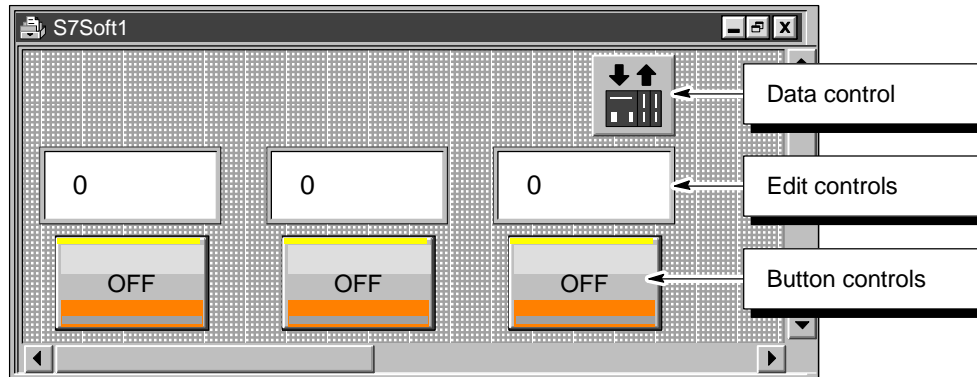


Figure 1-18 Using the SoftContainer to Create a Sample I/O Panel

Configuring the Properties for the SIMATIC Controls

You use the Properties dialog box of the Data control to connect the other SIMATIC controls to the control engine.

To assign a variable (memory location in the control engine) to a SIMATIC control, select the Data control and click the right mouse button (“right-click”) to display the shortcut menu. From the shortcut menu, select the **Properties** menu command for the Data control to display the Properties dialog box.

Configuring the the Control Engine for the Data Control

This example presumes that you have installed a control engine. For more information about connecting to control engines, see Section 5.3.

Use the following procedure to connect the Data control to the control engine:

1. Select the “Engine” tab of the Properties dialog box for the Data control.
2. Select the “Direct Connect” option and enter either **winLC** or **wcs7=3** (for a slot PLC such as the CPU 416-2 DP ISA) as the control engine.
3. Click on the “Apply” button to enter this data.

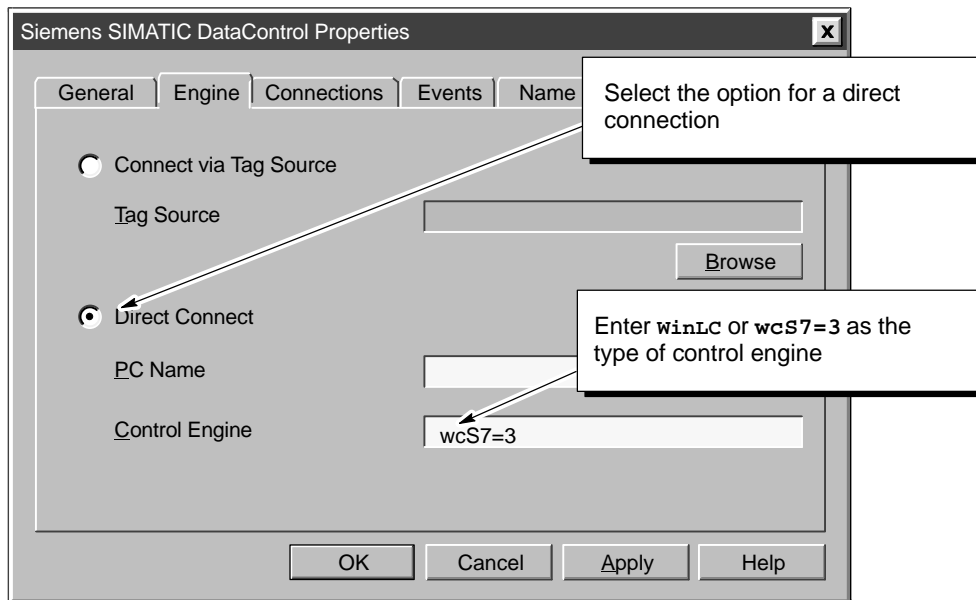


Figure 1-19 Connecting to the Control Engine (SoftContainer Example)

Assigning a Variable (Memory Location) to a Property

The Data control establishes a connection between the individual SIMATIC controls and the control engine. In the Properties Dialog box for the Data control, you assign variables (memory locations in the control engine) to the individual properties of the controls.

Refer to Table 1-2 and assign the variables (memory addresses in the control engine) to the SIMATIC controls.

Table 1-2 Assigning Sample Addresses to the SIMATIC Controls

Control	Address	Description
Edit1	QB0	Output value of first counter
Edit2	QB1	Output value of second counter
Edit3	QB2	Output value of third counter
Button1	I 0.0	Enable bit for first counter
Button2	I 0.1	Enable bit for second counter
Button3	I 0.2	Enable bit for third counter

Use the following procedure to connect the Value property of Button control Button1 to PI 0.0 in the control engine:

1. Select the "Connections" tab of the "Properties" dialog box for the Data control.
2. Click on the "+" beside SIMATIC Data1 (or double-click on SIMATIC Data1) to display the listing of the controls in the container.
3. Click on the "+" beside Button1 (or double-click on Button1) to display the properties for the Button control. See Figure 1-20.
4. Scroll down to and select (click on) the Value property. Notice that when you select the Value property, the "Assigned Variable" field becomes active.
5. As shown in Figure 1-20, enter "i0.0" in the "Assigned Variable" field. (You can use either upper case or lower case for designating memory locations.)
6. Click on the "Apply" button to enter the data.

Repeat this procedure for the other Button controls and the three Edit controls, entering the variables listed in Table 1-2. After you have configured the connections for all of the controls, click on the "OK" button to confirm the changes and close the "Properties" dialog box.

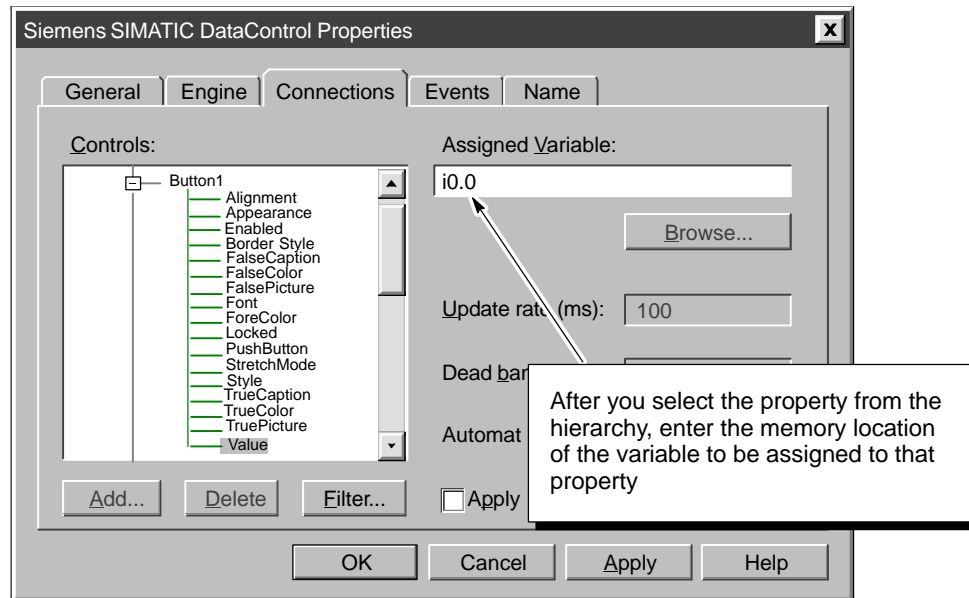


Figure 1-20 Assigning the Value Property to a Variable

Configuring the Edit Control for Binary Data

The Edit control is able to display data in a variety of formats. For this example, you will configure the Edit controls to display the byte of data (QB0, QB1, or QB2) in decimal format.

Note

The “Data type” field of the Edit control determines the size of the data to be displayed.

Use the following procedure to configure the Edit control:

1. Select the Edit control (Edit1) and click the right mouse button (“right-click”) to display the shortcut menu. From the shortcut menu, select the **Properties** menu command for the Edit control to display the “Properties” dialog box.
2. Click on the arrow beside the “Data Format” field to display the drop-down menu.
3. Scroll to the entry for decimal and click on “2 – wDecimal” to display the value in binary format (0 or 1). See Figure 1-21.
4. Click on the “Apply” button to enter the data and click on the “OK” button to close the “Properties” dialog box.

Repeat this procedure for the other Edit controls (Edit2 and Edit3).

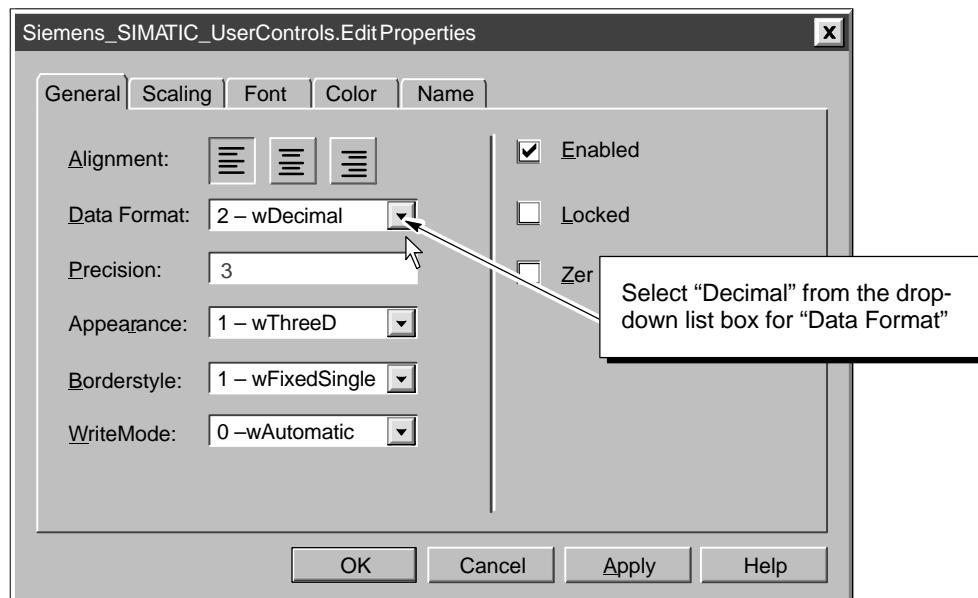


Figure 1-21 Configuring the Display Properties of the Edit control

Connecting the SIMATIC Controls to the Control Engine

If the control engine is not active, your controls have no process to monitor. When you are ready to use the controls to view or modify data, the control engine must be running.

Use the following procedure to connect the controls in the container to the control engine:

1. Click on the “Run” icon (or select the **Mode ► Run** menu command) to switch the container from Design mode to Run mode. See Figure 1-22.
2. Click on the Button controls to start (or stop) the counters in the sample program. Notice that when the Button control changes state, the value displayed in the corresponding Edit control changes.
3. Click on the “Design” icon (or select the **Mode ► Design** menu command) to switch the container from Run mode to Design mode (disconnecting the controls from the control engine).

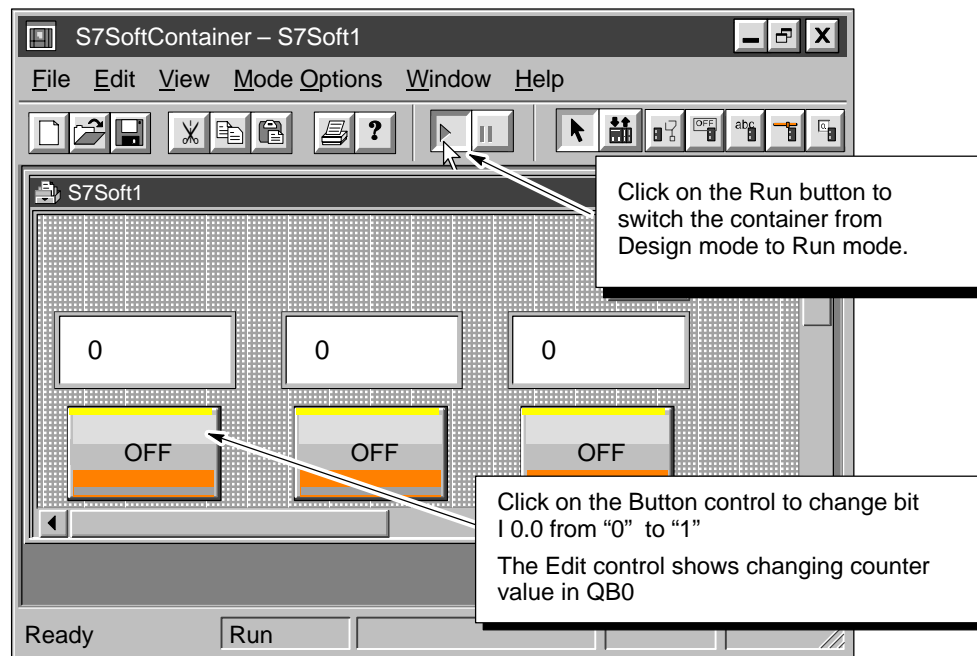


Figure 1-22 Placing the Container into Run Mode

Product Overview

2

Chapter Overview

The Computing package provides a method for other software applications to access the process data of your application. The Computing software provides ActiveX controls that can be inserted in any software application that is an ActiveX control container, like Visual Basic or Visual C++.

While the SIMATIC controls provided by Computing have been tested with other containers provided by other vendors, some third-party containers may not function as described in this document. Refer to Appendix F for guidelines about third-party containers and about using custom ActiveX controls with the Data control.



Warning

When you change the value that is displayed in an ActiveX control, whether from Computing or from a third-party software application, you are changing the value in your actual process.

Altering process data can cause unpredictable process operation, which could result in death or serious injury to personnel, and/or damage to equipment.

Exercise caution to ensure that you do not modify, nor permit unauthorized persons to access, any data that could cause process equipment to operate erratically. Always install a physical emergency stop circuit for your machine or process.

Section	Description	Page
2.1	Product Overview	2-2
2.2	Using an ActiveX Control to Access the Process Data	2-4
2.3	Connecting to Your Process with the OPC Server	2-6

2.1 Product Overview

The Computing software application allows you to access the control engine of your process to monitor and modify the process data. Figure 2-1 shows how the Computing software can be used with several control engines, such as the Windows Logic Controller (WinLC), a slot PLC such as the CPU416-2 DP ISA, or S7 systems. Depending upon the communications card in your PC, you can access S7 PLCs over an MPI, PROFIBUS-DP, or Industrial Ethernet network.

The Computing software allows you to use symbolic names—instead of absolute addresses—to access memory locations or control engines. These symbols are stored in a tag file, which is created automatically from the symbol table of the STEP 7 project.

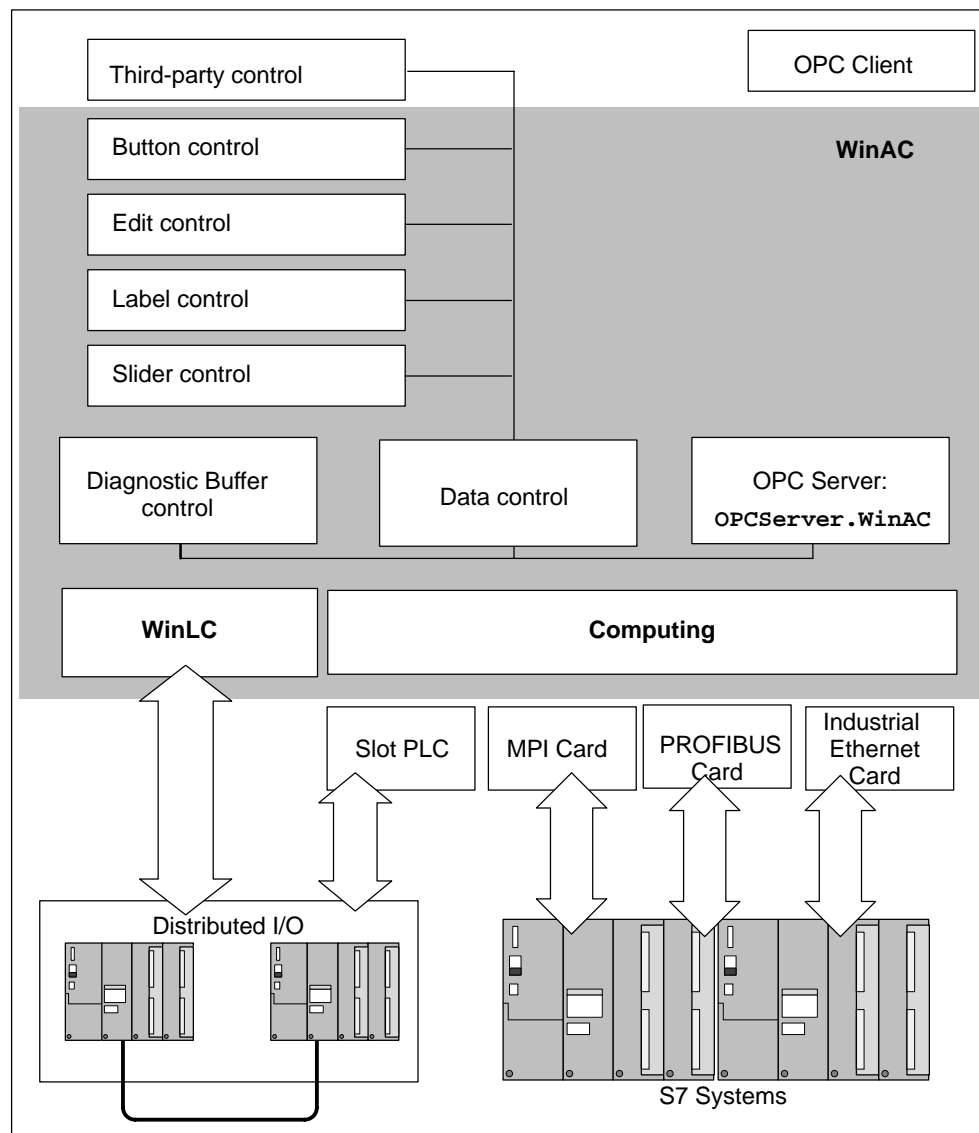


Figure 2-1 Accessing the Process Data with Computing

As shown in Figure 2-1, Computing provides several methods for accessing the process data:

- Computing provides standard ActiveX controls through the Data control that access the process data. You can use them with the Computing container provided by the Computing software, or you can insert these controls into containers of other software packages.
- Computing provides a diagnostics buffer of the S7 controllers. This buffer is a ring buffer that contains entries written by the operating system of the S7 controller. Each entry contains information about a specific diagnostic event. The DBuffer control allows your program to access the diagnostics buffer and display the events.
- Computing provides an OPC (OLE for Process Control) server that allows any OPC application to access data in the control device. Computing does not provide the OPC client application.

The OPC server is based on the OLE/COM technology from Microsoft. For more information about OPC, refer to the OPC specification: *OLE for Process Control Data Access Standard, version 2.0* from the OPC Foundation.

System Requirements

To run the Computing software, it is recommended that your computer meet the following criteria:

- A personal computer (PC) with the following:
 - Pentium processor running at 166 MHz or faster (recommended)
 - 64 Mbytes RAM
 - Microsoft Windows NT version 4.0 (or higher) with Service Pack 3
- A color monitor, keyboard, and mouse (or other pointing device) which are supported by Microsoft Windows NT
- A hard drive with 20 Mbytes of free space
- At least 1 Mbyte free memory capacity on drive C for the Setup program (Setup files are deleted when the installation is complete.)

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

2.2 Using an ActiveX Control to Access the Process Data

Computing provides access through the Data control to the process data being controlled by a control engine such as WinLC (the Windows Logic Controller). You can use the standard SIMATIC controls provided with the Computing software (see Table 2-1), or you can connect any other ActiveX control to the Data control.

Computing does not allow you to write data to timers. You can only read the timer values.

Table 2-1 Standard Controls Provided by Computing

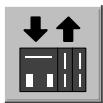
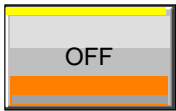



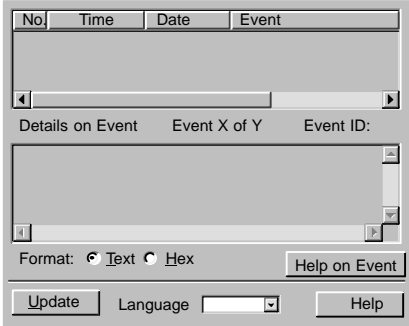
Control	Representation	Description
Data		Provides the connection to the control engine (for example, WinLC). Without the Data control, none of the other controls have access to the process data.
Button		<p>Provides access to individual memory bits in the control engine. The Button control accesses only bits and has two values:</p> <ul style="list-style-type: none"> Off = 0 (default color: red) On = 1 (default color: green) <p>Changing the state of the Button control changes the state of the variable in your process that is associated with the control.</p> <p>If you configure the Button control to be read-only, then it functions like a lamp or LED.</p> <p>If you configure the button to be a pushbutton, then it functions like a toggle switch</p>
Edit		<p>Provides access to memory locations in the control engine. You can access bytes, words, or double-words, and you can manipulate individual bits of this data.</p> <p>Entering a new value in the Number control changes the data in the control engine.</p>
Label		The Label control allows you to display a constant string. It is also possible to connect the Caption property of the Label control with any process value. The process value is converted into a string and displayed.
Slider		<p>Provides access to memory locations in the control engine. You can access bytes, words, or double-words.</p> <p>Adjusting the value of the Slider control changes the data in the control engine.</p>

Table 2-1 Standard Controls Provided by Computing, continued

Control	Representation	Description
DBuffer (S7 Diagnostics Buffer)		<p>Displays the diagnostics buffer of the controller.</p> <p>The DBuffer control connects directly to the controller: it does not use the Data control to make the connection.</p>



Caution

Using the timer function improperly or using breakpoints in Visual Basic with Computing can cause problems that could potentially cause your computer or application to crash or lock up. Depending on the configuration, this could cause the application to lose communication with the control engine. This could cause unpredictable process operation, which could result in death or serious injury to personnel, and/or damage to equipment.

Always install a physical emergency stop circuit for your machine or process.

Concerning VB timers: The Timer function in Visual Basic version 5 allows a timer to interrupt code in progress within the same thread, which can cause problems with potentially serious consequences. If you use VB timers with Computing, observe the following guidelines:

- Always kill (disable) the timers in the Form_Unload subroutine. Otherwise, a timer can trigger an event while the VB program is shutting down; this condition could cause your computer or your application to crash, lock up, or to continue running invisibly.
- If you start your timer in the Form_Load subroutine, the timer event could occur before the other objects have been instantiated. In order to ensure that the objects have been properly instantiated, always start a timer in the Form_Load subroutine with a large interval (such as 1 or 2 seconds) to allow the objects to be properly instantiated. Subsequent timer intervals can be set to shorter intervals.

2.3 Connecting to Your Process with the OPC Server

OLE for Process Control (OPC) provides a standard mechanism for communicating to numerous data sources, whether they be the devices on your factory floor or a database in your control room. You can use the OPC server provided with the Computing software to communicate with the control engine (for example, the WinLC controller) and provide access to the process data. Computing provides an OPC server that allows any OPC client application to access data in the control engine; Computing does not provide any OPC client application.

Computing implements only the mandatory interfaces as defined in the version 2.0 specification from the OPC Foundation. The interfaces defined in that specification as “custom” may be implemented at a later date.

OPC is based on the OLE/COM technology from Microsoft. For more information about OPC, refer to the OPC specification *OLE for Process Control Data Access Standard, version 2.0* from the OPC Foundation.

Connecting Computing to Client Applications

OPC allows you to access data from the plant floor and integrate the data into your existing business systems. You can use off-the-shelf tools (such as SCADA packages, databases, spreadsheets) to assemble a system that meets your needs. As shown in Figure 2-2, OPC provides an open and effective communication architecture which concentrates on data access and not the types of data.

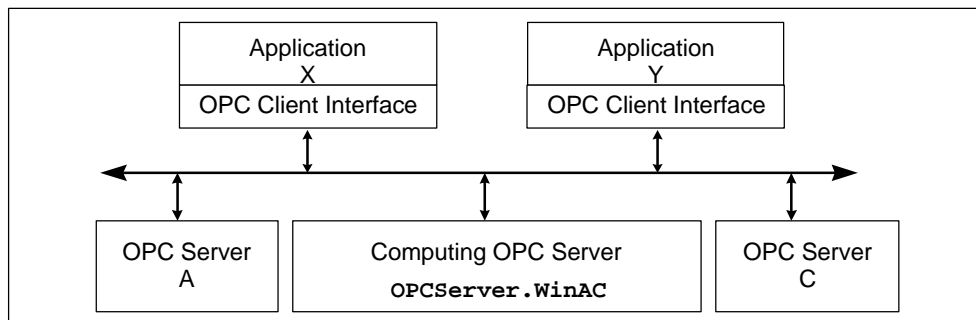


Figure 2-2 Applications Working with Many OPC Servers

Your OPC client connects to the OPC server object provided by Computing. This connection allows you to create and manipulate OPC group objects, which organize the data to be accessed. You can activate or deactivate a group as a unit, or you can “subscribe” to the list in a group of items so that you can be notified when the data change. (A group is a collection of items, like MB0.) Figure 2-3 shows the connection from the OPC client application through WinAC to the process data.

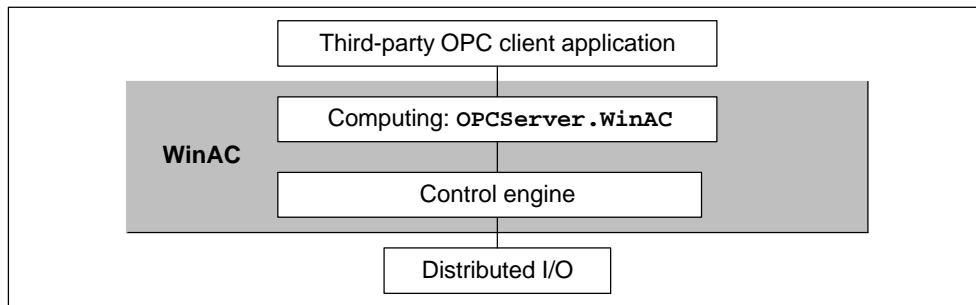


Figure 2-3 Using the OPC Server to Access Your Process Data

To access the OPC server and its contents, you must provide your OPC client with the name (ProgID, or programmatic identifier) of the server object. The name of the OPC server object that is provided by Computing is: `OPCServer.WinAC`

For more information about the OPC server provided by Computing, refer to the *OPC Server Interface Manual*.

Setting Up Computing Software

3

Chapter Overview

This chapter provides the following information:

- Section 3.1 lists the authorization requirements for installing and running the Computing software.
- Section 3.2 describes how to install the authorization.
- Section 3.3 describes how to install and uninstall the Computing software.
- Section 3.4 describes how to use the PG/PC Interface to connect Computing to a slot PLC or a communications card.

Section	Description	Page
3.1	Overview	3-2
3.2	Authorization	3-3
3.3	Installing and Uninstalling the Computing Software	3-5
3.4	Connecting Computing to a Slot PLC or Communications card	3-7

3.1 Overview

The Computing software provides ActiveX controls, which you can use to create a tailored view into your process. Computing lets you use any mix of S7 and third-party ActiveX controls not only to view, but also to modify, process data.

Follow the guidelines below for authorization for your Computing software.

- For WinAC Basis, use the WinAC authorization
- For SIMATIC Net, use the SIMATIC Net authorization
- For upgrading WinAC Pro with Computing 3.0 standalone, use the Computing authorization
- For standalone, use the Computing authorization.

Note

To run Computing on a different PC than the WinLC, you must purchase SIMATIC Computing standalone.

3.2 Authorization

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

Note

If you remove the authorization, Computing continues to operate; however, a notification message appears every six minutes to alert you that the authorization is missing.

Authorization Disk

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

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



Caution

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

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

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 steps to install the authorization for Computing.

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 Computing.

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

Removing an Authorization

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

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

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

You can then use the diskette again to install an authorization. You must use the authorization diskette to remove any existing authorizations. If you need to remove Computing completely, you must remove the DP authorization.

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

3.3 Installing and Uninstalling the Computing Software

The Computing software includes a Setup program that executes the installation automatically. Prompts on the screen guide you step by step through the installation procedure.

Note

You must have administrator ("ADMIN") privileges to install the Computing software.

Starting the Installation Program

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

1. Insert the CD-ROM in your computer.
2. Use the Windows NT Start menu (select the **Start ► Run** menu command) to open the "Run" dialog box.
3. Click on the "Browse" button on the "Run" dialog box and select the installation program (Setup.exe) on the CD-ROM.
4. Click on the "Open" button to enter the Setup.exe program into the "Run" dialog box.
5. Click on the "OK" button to start the installation program.
6. Follow the instructions displayed by the installation program.

To install only the Computing software, deselect the other components.

7. When prompted by the software, enter the registration number.

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

If an Older Version of Computing Is Already Installed

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

- Abort the installation so that you can uninstall the older version of the Computing software under Windows NT and then start the installation again.
- Continue the installation and overwrite the older version with the new version.

Your software will be better organized if you uninstall any older versions before installing the new version. Overwriting an old version with a new version has the disadvantage that if you then uninstall, any remaining components of the old version are not removed. If you uninstall the older version of Computing, you must reboot your computer before installing the new version.

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 20 Mbytes of free space on your hard disk.
- Bad disk: Verify that the disk is bad, then call your local Siemens representative.
- Operator error: Start the installation again and read the instructions carefully.

Uninstalling the Computing Software

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

1. Start the dialog box for installing software under Windows NT by double-clicking on the "Add/Remove Programs" icon in the Control Panel.
2. Select the Computing entry in the displayed list of installed software. Click on the "Add/Remove..." button to uninstall the software.

3.4 Connecting Computing to a Slot PLC or Communications Card

To connect Computing to a slot PLC or communications card, you must define the network connection over which Computing and the PLC or card communicate by setting the PG/PC Interface.

Note

You can only view one slot PLC or one communications card at a time.

Follow these steps to configure Computing for communicating with a slot PLC or communication card:

1. From the Windows NT Start menu, select **Start ► Simatic PC Based Control ► WinCP Configurator**.
2. Select the the Connection tab, then click on the Setting the PG/PC Interface button. The PG/PC Interface dialog box appears.

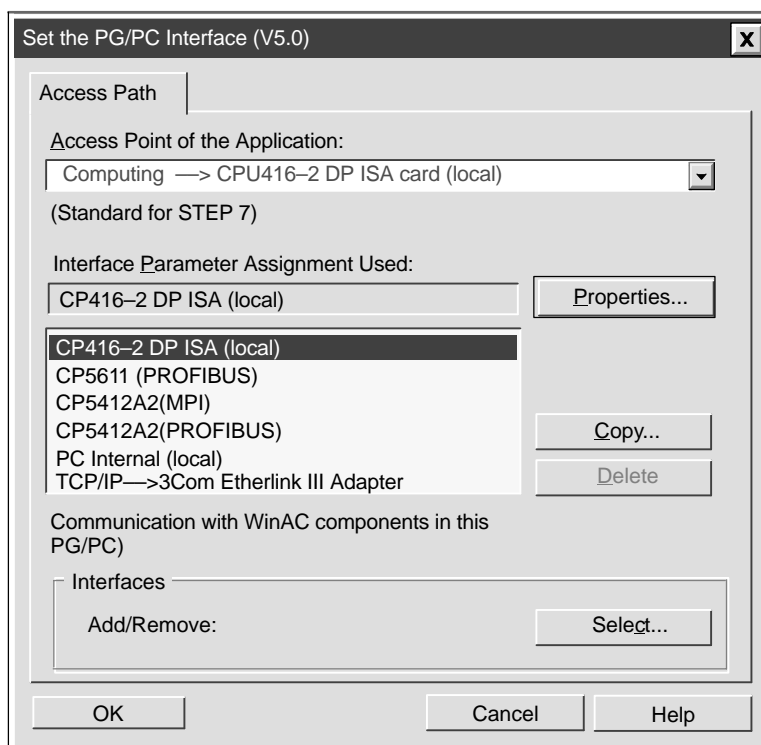


Figure 3-1 Setting the PG/PC Interface for Slot PLC CPU 416-2 DP ISA (local)

3. From the “Access Point of Application” drop-down list, select **Computing**.
4. Select the interface parameter from the parameter set that corresponds to your network communications path, for example “CPU416-2 DP ISA (local)”. The PLC or card you selected appears in the Access Point of the Application field (Figure 3-1).

To set the Computing interface setting for a local slot PLC, select:

COMPUTING → <cardname> (local)

To set the Computing interface setting for an S7 system on a TCP/IP LAN, select:

COMPUTING → <cardname> (TCP/IP)

To set the Computing interface setting for an S7 system on an Industrial Ethernet network (ISO Transport protocol), select: **COMPUTING → <cardname> (ISO Transport)**

To set the Computing interface setting for an S7 system on a PROFIBUS network, select **COMPUTING → <cardname>(PROFIBUS)**.

Using Computing to Access Data

4

Chapter Overview

Computing allows you to access data from control engines (either WinLC of WinAC Basis, a slot PLC such as the CPU 416–2 DP ISA of WinAC Pro, or other S7 PLCs. These control engines can be installed on the same computer as Computing, or Computing can use the local area network to access the control engine.

Note

The term “slot PLC” in the manual refers to a slot PLC such as CPU 416–2 DP ISA or CPU 416–2 DP ISA Lite. In the manual, the CPU416–2 DP ISA Lite is treated identically to the CPU416–2 DP ISA.

Computing allows you to communicate across a local area network using Windows NT's Distributed Component Object Model (DCOM). You use DCOM to integrate distributed applications by way of a LAN. A distributed application consists of multiple processes or different computers that cooperate to accomplish a single task.

Computing can also communicate with a local or remote control engine in an MPI, PROFIBUS–DP, or H1 network. You can use the PG/PC Interface to set up connection with the control engine.

Section	Description	Page
4.1	Accessing Data in Control Engines	4-2
4.2	Accessing a Local Control Engine	4-4
4.3	Accessing a Remote Control Engine	4-5
4.4	Communicating with Multiple Control Engines	4-6

4.1 Accessing Data in Control Engines

A “control engine” is a processor or program that manages and manipulates data which is used to control a process or machine. The control engine can be either software or hardware. As shown in Figure 4-1, the elements of the Computing software allow you to access data in the following control engines:

- Windows Logic Controller (WinLC), a software-based S7 controller that runs on the RAM of your computer. WinLC communicates with I/O modules over a PROFIBUS–DP network. With release 3.0, SIMATIC Computing can also access data in WinLC over PROFIBUS–DP, MPI, and H1 networks.
- A slot PLC such as the CPU 416-2 DP ISA, an S7 controller built on an ISA card that is installed in your computer. This CPU communicates with I/O modules over a PROFIBUS network.
- Other S7 CPU modules on an MPI, H1, or PROFIBUS network. These CPUs have local and distributed (remote) I/O.

Computing provides SIMATIC controls that utilize Microsoft’s ActiveX technology to allow third-party applications (such as Microsoft’s Excel or Visual Basic) to access data in the control engine. In addition to these SIMATIC controls, Computing provides a server that allows other applications to use an OPC (OLE for Process Control) interface.

Using a tag file, you can access the data symbolically on a PC where STEP 7 is not installed. The TagFile Configurator creates a tag file from the symbol table of STEP 7. The tag file also provides you with the capability to connect your application to more than one control engine simultaneously. (See Section 4.4.)

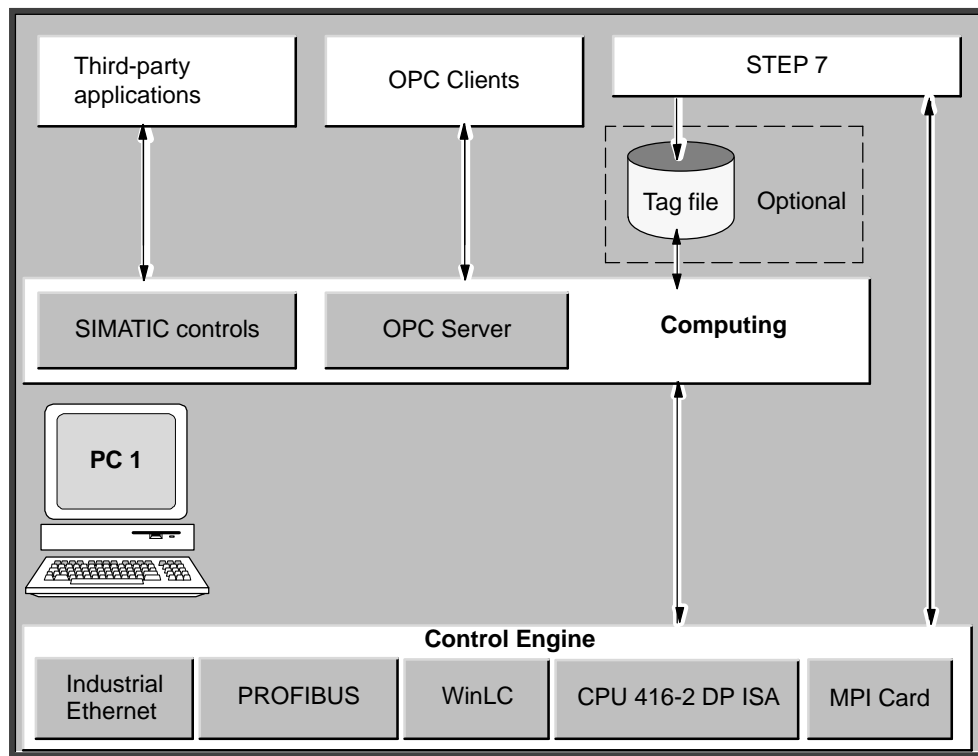


Figure 4-1 Accessing Data in Control Engines

4.2 Accessing a Local Control Engine

The basic configuration for Computing is to have all of the elements running on one computer, as shown in Figure 4-2. All of the software runs on the local computer, and access between the applications is simplified.

Using a tag file, you can access the data in the control engine by using symbols instead of absolute addresses. As shown in Figure 4-2, your application program could use the symbol “Fill_Valve” to access I 0.0 and the symbol “Drain_Valve” to access I 0.1 in the control engine. You use the symbol table in STEP 7 to create the tag file—the same symbol table that you created when you created the program for the control engine.

The tag file also provides you with the capability to connect your application to more than one control engine simultaneously. For information about connecting to multiple control engines, see Section 4.4.

For more information about creating tag files, see Chapter 9.

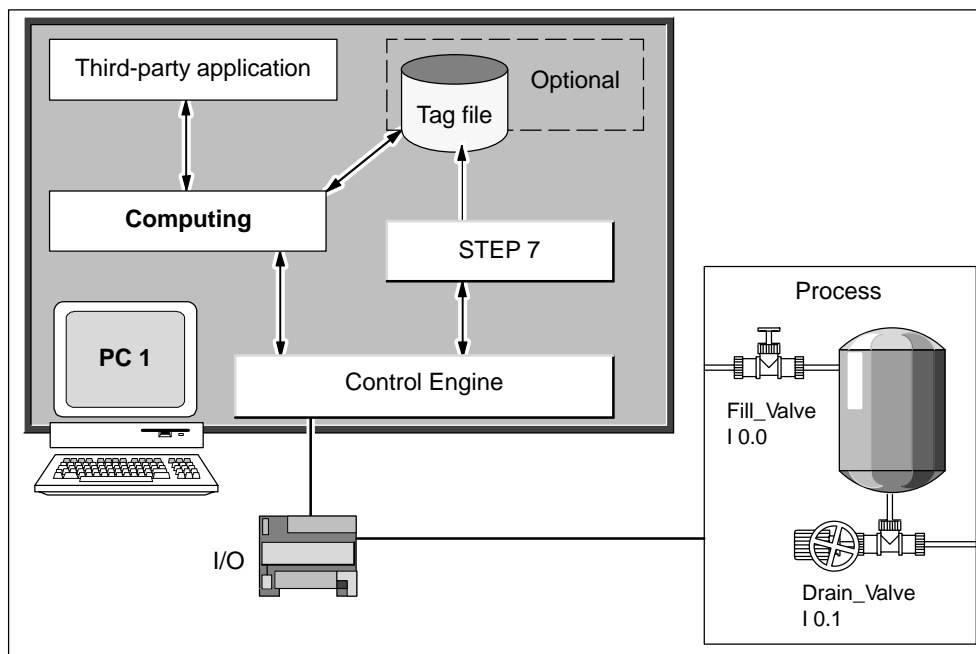


Figure 4-2 Accessing a Local Control Engine

4.3 Accessing a Remote Control Engine

You can also use Computing to access control engines on remote computers that are part of your local area network (LAN). As shown in Figure 4-3, a third-party or custom application which is running on PC 2 can access data in the control engine which is running on PC 1.

Using a tag file, you can access the data in the control engine by using symbols instead of absolute addresses. As shown in Figure 4-2, your application program could use the symbol “Fill_Valve” to access I 0.0 and the symbol “Drain_Valve” to access I 0.1 in the control engine. You use the symbol table in STEP 7 to create the tag file—the same symbol table that you created when you created the program for the control engine.

To use symbols with the application running on the remote computer, you must copy the tag file to the remote computer. The remote computer does not require STEP 7 or the symbol table to use the tag file.

For more information about creating tag files, see Chapter 9.

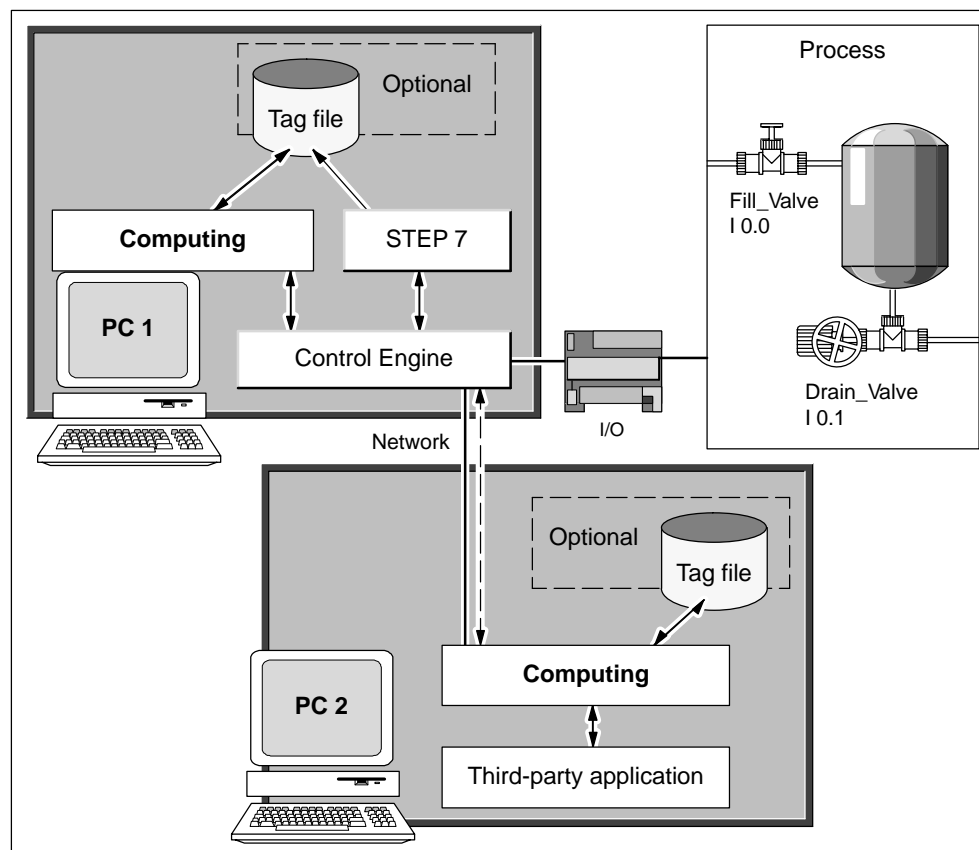


Figure 4-3 Accessing a Remote Control Engine

4.4 Communicating with Multiple Control Engines

You can use Computing over a local area network to access several control engines simultaneously. By creating a tag file, you can configure the Data control to access data on multiple control engines.

Note

You can only view one slot PLC or one communication card at a time

By utilizing Microsoft's DCOM technology and/or other Siemens networks, a tag file allows you to access a variety of control engines and applications throughout your local area network. As shown in Figure 4-4, a computer running Computing (PC 1) can use a tag file to access data in the control engines running on another computer (PC 2), as well as a third-party application (such as Microsoft's Excel or Visual Basic).

As with a single control engine (see Sections 4.2 and 4.3), the tag file also provides the capability for using symbols to access the data in the various control engines.

See Appendix E for information about using DCOM to connect Computing to multiple control engines. For more information about tag files and the TagFile Configurator, see Chapter 9.

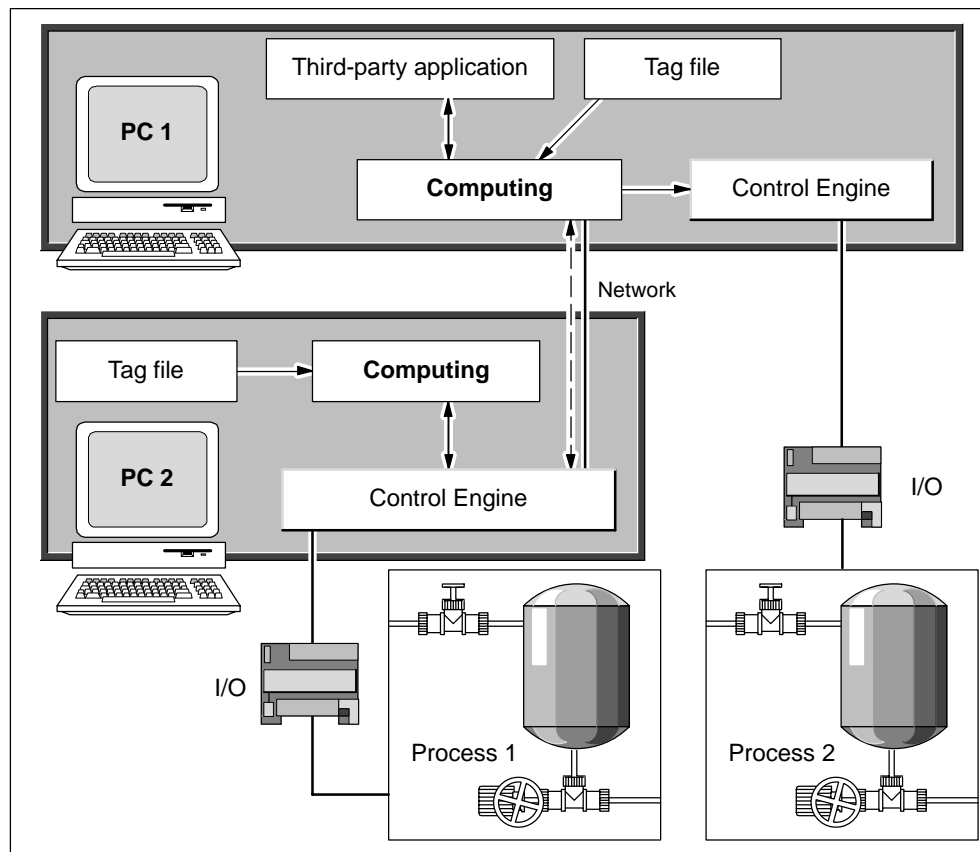


Figure 4-4 Accessing Data in Several Remote Control Engines

Accessing the Process Data with the Data Control

5

Chapter Overview

The Data control provides the connection between your ActiveX controls and the control engine (for example, WinLC or a slot PLC such as the CPU 416-2 DP ISA).

The Data control has specific properties that you can configure:

- For the SoftContainer provided by SIMATIC Computing: double-clicking on the Data control displays the Properties dialog box. This dialog box contains the following tabs: General, Engine, Connections, Events, and Name.
- For other container applications (such as Microsoft Visual Basic): access the properties as for any other control in that container (for example, by using the right mouse button). Open the context menu for the Data control by clicking the right mouse button and selecting the **Properties** menu command.

Section	Description	Page
5.1	Connecting the SIMATIC Controls to the Control Engine	5-2
5.2	Configuring the Connection Properties for the Data Control	5-3
5.3	Selecting the Control Engine for the Data Control	5-4
5.4	Connecting the ActiveX Controls to the Control Engine	5-9
5.5	Filtering the Properties for the ActiveX Controls	5-13
5.6	Configuring Custom Events	5-15
5.7	Creating a Connection Table	5-16
5.8	Sample Program for Creating a Connection Table and an Event Table	5-17
5.9	Sample Program for Responding to Events	5-19
5.10	Sample Program for Reading and Writing Data	5-24
5.11	Sample Program for Reading and Writing Boolean Data	5-29
5.12	Properties, Methods, and Events of the Data Control	5-30

5.1 Connecting the SIMATIC Controls to the Control Engine



Caution

Failing to disable the timers in your program could cause timer-generated connections to remain connected, allowing these connections to continue to write data to the control engine. This could cause the control engine to operate erratically, which could potentially cause damage to equipment and injury to personnel.

To ensure that all connections are disconnected when your program closes, always disable all timers before the End statement in the Form_Unload subroutine.

In order to access process data, the SIMATIC Computing controls (Button, Edit, Label, and Slider) must first establish a connection through the Data control. Figure 5-1 shows the relationship between the Data control and the other SIMATIC controls.

Note

The Panel control (available with WinLC or a slot PLC) and the DBuffer control (for accessing the diagnostics buffer of S7 controllers) do not use the Data control for connecting to the control engine.

You use the “Connections” tab of the Data control to assign a variable (the memory location) to the Value property of each control. The Data control configures the control engine to check the memory locations of the assigned variables at a specified rate (in milliseconds). If there is a change in the value, the new value is written to the Data control. The Data control then writes the new value to the other controls.

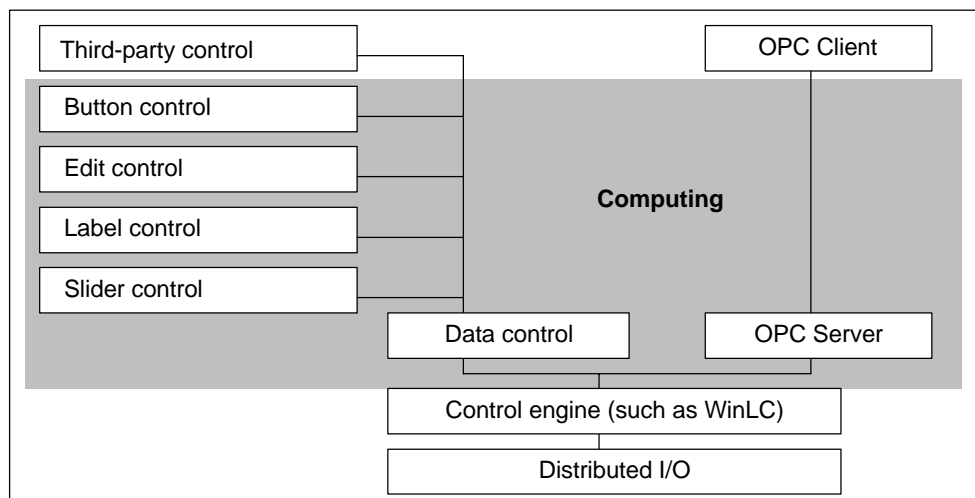


Figure 5-1 Using the Data Control for Connecting to a Control Engine

5.2 Configuring the Connection Properties for the Data Control

As shown in Figure 5-2, the “General” tab allows you to configure the following parameters for the connection to the control engine:

- **AutoConnect** (automatic connection): when this option is enabled, the Data control automatically connects to the memory locations in the control engine. When this option is disabled, the Data control connects to the memory locations only when instructed by the program code (using a Connect method) that you associated with the control.
- **AutoConnect Timeout**: time-out (in milliseconds) for the automatic connection: specifies the amount of time that the Data control waits between connecting to the control engine and writing data.

Some containers may not provide a mechanism for telling the Data control to write to the control engine. After the time-out that you specify, the Data control starts writing data.

- **Default Update Rate (ms)**: specifies the rate (in milliseconds) that the control engine checks the memory locations to see if change has occurred.
- **Default Dead Band**: specifies to the control engine the amount of change that must occur in a value before the control engine writes the new value to the Data control. For example: if the dead band is 10 and the value in the control engine is 22, then the control engine does not write a new value until the value becomes either 33 or 11.
- **Show Error Boxes**: specifies whether to display the default error boxes when there is a user-generated error. Computing provides error messages in English only. If you want to display messages in other languages, you must deselect this option and write program code to react on the error event.

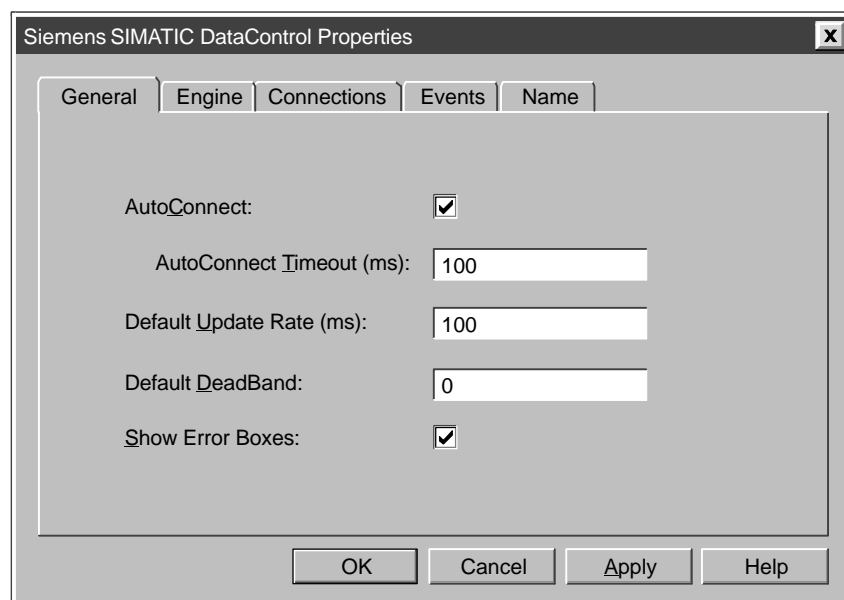


Figure 5-2 Data Control Properties (General Tab)

5.3 Selecting the Control Engine for the Data Control

Computing allows you to connect either to a single control engine or to several control engines. You can also connect to the control engine over a network, such as a local area network (LAN). Use the “Engine” tab (Figure 5-3) of the “Properties” dialog box to select the control engine.

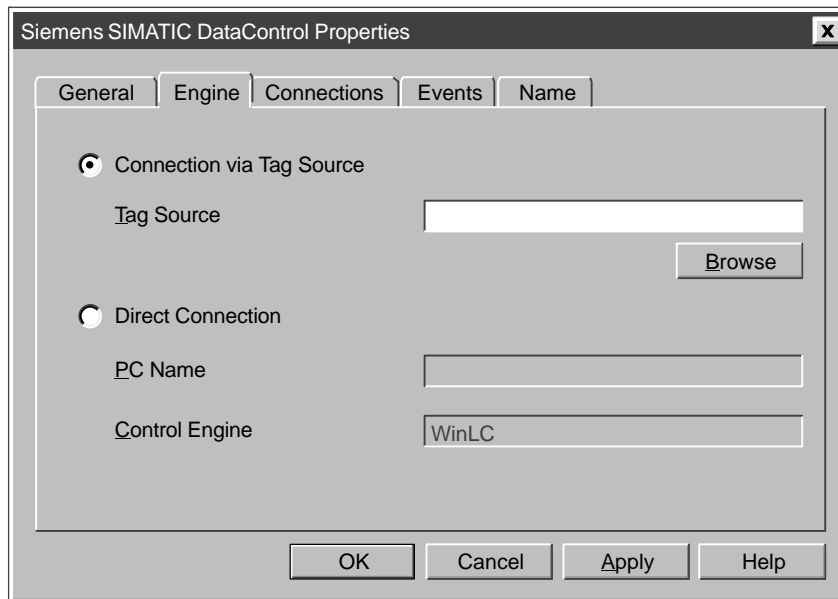


Figure 5-3 Data Control Properties (Engine Tab)

A control engine may be either a slot PLC, such as the CP 416-2 DP ISA, an S7 CPU, or WinLC. The control engine strings for the Data Control are of two types:

- Direct to WinLC
- Over a Siemens network

The control engine string for a direct connection to WinLC is **WinLC**. It is a COM connection; you do not need to use the PG/PC Interface to configure it.

Connections for the following network types are configured using the PG/PC Interface settings are as follows:

- access to MPI network: COMPUTING—> <cardname> (MPI)
- access to DP network: COMPUTING—> <cardname> (PROFIBUS)
- access to H1 network: COMPUTING—> TCP/IP—> <cardname> (For access to the network, the NCM option package and STEP 7 v5 SP3 must be installed.)

Note

To use WinLC as a control engine, you must also set up the connection in WinLC. Refer to the chapter on connecting STEP 7 to WinLC or PLCs in the *SIMATIC Windows Logic Controller (WinLC) Manual* for directions.

Using a Tag File

The tag file (or tag source) provides symbolic identification that can be used by the Data control to access data or control engines. Using a tag file provides the following capabilities:

- You can connect the Data control to several different control engines at the same time.
- You can use the symbols defined in STEP 7 to access memory locations in the control engine.

The tag file also provides a completion aid for entering the symbols. For more information about tag files, see Chapter 9.

For more information about connecting to multiple control engines, see Section 4.4. See Section 9.1 for information about using STEP 7 and the TagFile Configurator for connecting to remote control engines.

Use the following procedure to connect the Data control to the control engine which is specified in a tag file:

1. Double-click on the Data control (or use the **Edit ► Properties** menu command) to display the “Properties” dialog box for the Data control.
2. Click on the “Engine” tab to display the configuration choices.
3. Select the “Connection via Tag Source” option.
4. As shown in Figure 5-4, click on the “Browse” button and select the valid tag file (*.tsd).
5. Click on the Apply button to configure the Data control for using the specified tag file for connecting to the control engine.

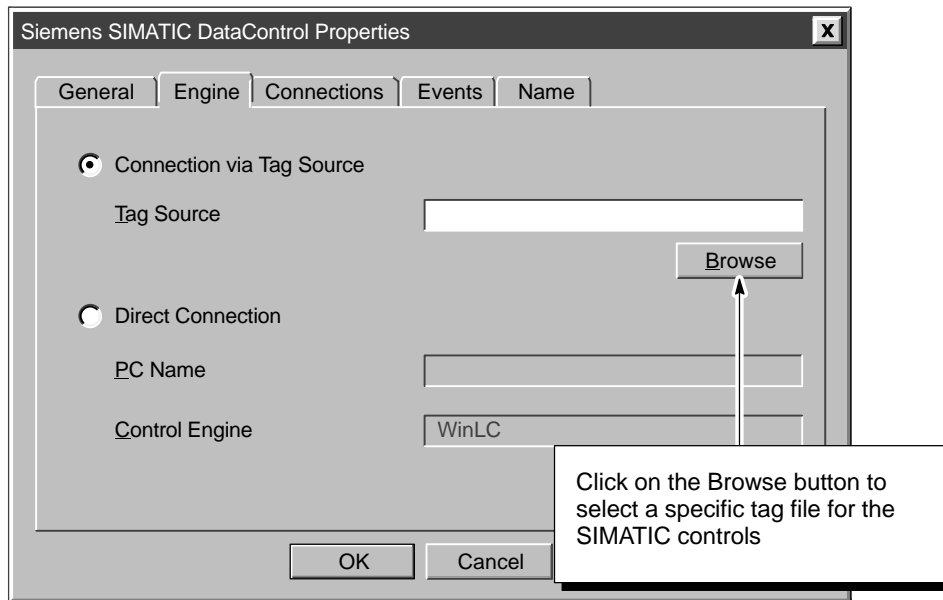


Figure 5-4 Selecting a Tag File for the Data Control

Connecting to a Specific Control Engine

As shown in Figure 5-5, you can use the Data control to connect your program to a control engine residing either on a local computer or on a different computer.

If you are connecting to a control engine over a LAN via DCOM, you must also specify the network name of the server PC in the “PC Name” field.

If you select the “Direct Connection” option, you must specify the name of the control engine in the control engine field. The control engine strings now support rack and slot location of the CPU at the network node, and include H1 networks, both IP and MAC addresses. Enter the following strings in the “Control Engine” field:

- **<local>** (to specify the control engine located on the same computer as the Computing software)
- **WinLC** (to specify the WinLC of WinAC Basis)
- **wcs7=3** (to specify a slot PLC such as the CPU 416-2 DP ISA of WinAC Pro)
- **wcs7=xx,a,b** (to specify other PLCs on the MPI network, where xx is the MPI address, a is the rack number, and b is the slot number.) Note that WinLC is always assigned rack 0, slot 2.
- **wcIP=xxx.xxx.xxx.xxx,a,b** for a control engine on a TCP/IP LAN or **wcMAC=xx.xx.xx.xx.xx,a,b** for a control engine on an Industrial Ethernet network with STEP 7 v5 SP3. Refer to Appendix G for more information about control engine strings.

Some strings have been simplified for easier entry, but Data Control will accept longer strings used by applications that were created with previous versions of Computing (for example, S7DosIntfMPI=3).

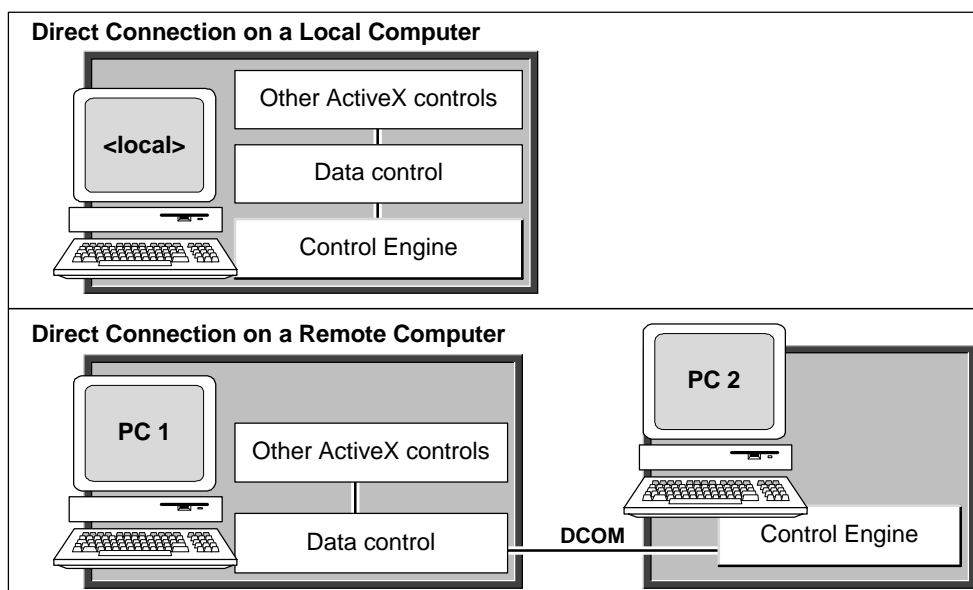


Figure 5-5 Direct Connection for a Local or a Remote Computer

Note

When you configure the Data control to connect to a single (specific) control engine, you cannot connect a tag file. This means that you cannot use symbol names for the variables in the control engine.

To use symbol names, select the option for connecting to multiple control engines and browse to a tag file that contains symbols for only one control engine.

Use the following procedure to configure the Data control for connecting to a specific control engine:

1. Double-click on the Data control (or use the **Edit ► Properties** menu command) to display the "Properties" dialog box for the Data control.
2. Click on the "Engine" tab to display the configuration choices.
3. Select the "Direct Connection" option. See Figure 5-6.
4. To connect to a control engine on the local computer, enter **<local>** in the "PC Name" field. Note that you cannot access WinLC and a slot PLC simultaneously.

To connect to a control engine on a remote computer:

- In the "PC Name" field, enter the network name of server computer (for example, "PC_2").
- In the "Control Engine" field, enter the name of the control engine, such as **WinLC** or **wcs7=3** (for a slot PLC such as the CPU 416-2 DP ISA of WinAC Pro).

5. Click on the “Apply” button to configure the Data control.

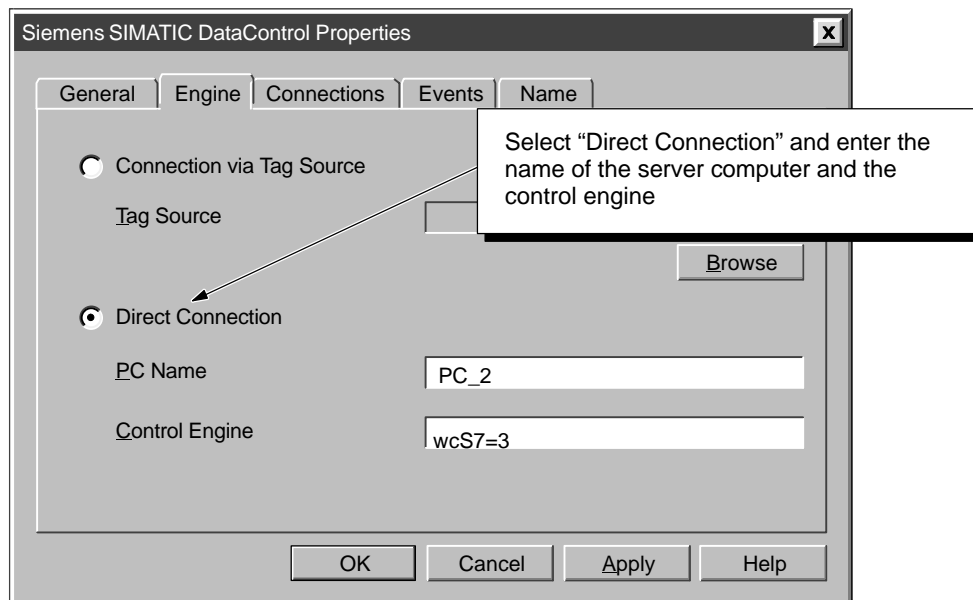


Figure 5-6 Configuring DCOM for a Specific Control Engine

Note

To connect to a remote computer via LAN, you must have configured the different computers for DCOM. See Sections E.2 and E.3 for information about configuring the server and client computers for DCOM.

5.4 Connecting the ActiveX Controls to the Control Engine

The Connections tab shows the ActiveX controls (whether they are SIMATIC controls or third-party) that can be connected to the control engine.



Caution

Using the timer function improperly or using breakpoints in Visual Basic with Computing can cause problems that could potentially cause your computer or application to crash or lock up. Depending on the configuration, this could cause the application to lose communication with the control engine. This could cause unpredictable process operation, which could result in death or serious injury to personnel, and/or damage to equipment.

Always install a physical emergency stop circuit for your machine or process.

Concerning VB timers: The Timer function in Visual Basic version 5 allows a timer to interrupt code in progress within the same thread, which can cause problems with potentially serious consequences. If you use VB timers with Computing, observe the following guidelines:

- Always kill (disable) the timers in the Form_Unload subroutine. Otherwise, a timer can trigger an event while the VB program is shutting down; this condition could cause your computer or your application to crash, lock up, or to continue running invisibly.
 - If you start your timer in the Form_Load subroutine, the timer event could occur before the other objects have been instantiated. In order to ensure that the objects have been properly instantiated, always start a timer in the Form_Load subroutine with a large interval (such as 1 or 2 seconds) to allow the objects to be properly instantiated. Subsequent timer intervals can be set to shorter intervals.
-

Assigning a Variable to a Property of a Control

To define a connection in the control engine, you assign a variable (memory location) in the control engine to a property of a control. See Figure 5-7.

Note

Computing does not allow you to write to timers in the control engine.

Use the following procedure to assign a variable in the control engine to a property of the control:

1. On the Connections tab of the Data control, select (click on) the name of the property.

2. In the “Assigned Variable” field, enter the memory address in the control engine:

Note

If you have attached a tag file to the Data control, you can enter the symbols instead of the absolute addresses. You can also click on the “Browse” button to navigate to the symbol. Entering a “.” (dot or period) displays a list of valid tag files or symbols of each hierarchic level.

- If you selected the “Direct Connection” option for the control engine (not using symbols), enter the absolute address (such as **MB0**) for the memory location in the control engine. See Appendix A for information about the data types and memory areas of the S7 controllers.
 - If you selected the “Connection via Tag Source” option to use multiple control engines (but are **not** using symbols), and the absolute address (such as **MB0**) for the memory location in the default control engine. See Appendix A for information about the data types and memory areas of the S7 controllers. You can also append an absolute variable address to the symbolic name of a control engine (for example, **ce1.mb0**, **ce2.mb0**). See “Using Absolute Addresses with a Tagfile” in Section 9.3 for information about setting the default control engine.
 - If you selected the “Connection via Tag Source” option to use symbols (regardless of whether you are connecting either to a single control engine or to multiple control engines) enter the symbolic address for the memory location (such as **Start_Program**). You can append the address to the symbolic name of the control engine (such as **PC_2_WinLC**).
3. Click on the “Apply” button to assign the variable to the property.

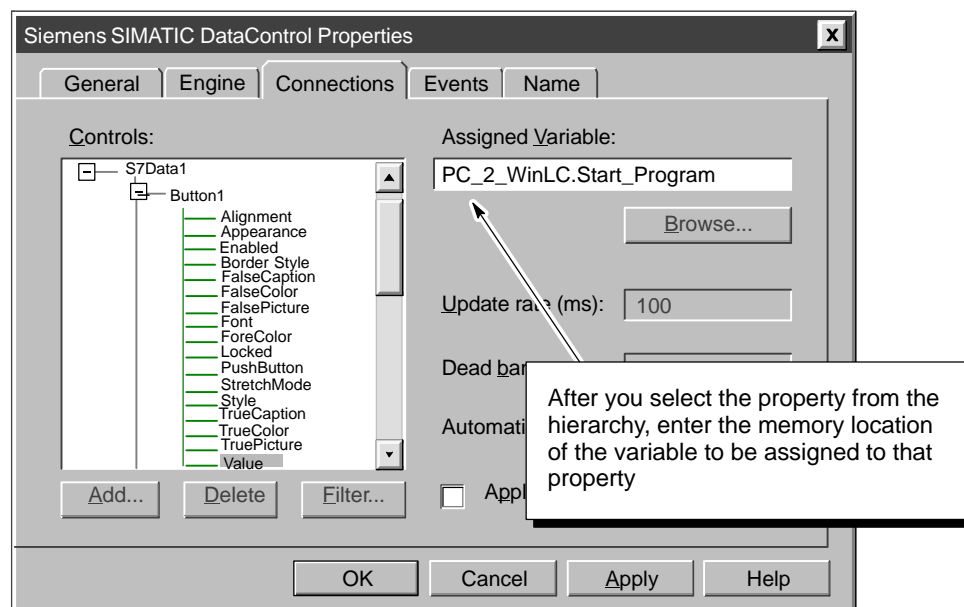


Figure 5-7 Entering a Symbol for the Assigned Variable

If you identified a tag source or file, you can browse to a symbol for the variable. Click on the “Browse” button (as shown in Figure 5-8) and select the variable from the symbols listed for the tag source or file. Select a symbol or tag and press Enter. (Figure 5-8 shows the property listing with a filter applied to display only the Enabled and Value properties. for more information about filtering the property listing, see Section 5.5.)

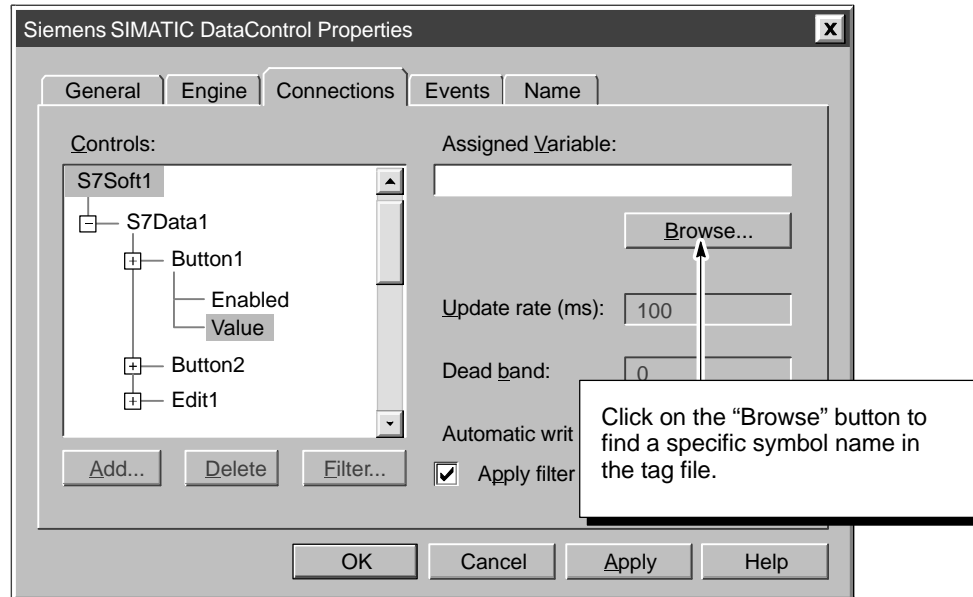


Figure 5-8 Browsing to a Symbol in the Tag File

Entering Absolute Addresses Instead of Symbols

If you attached a tag file to the Data control, you can enter absolute addresses instead of the symbols created by the symbol table of STEP 7. Click on the Browse button to navigate to the symbol. Entering a character in the Assigned Variable field displays a list of valid symbols at the first level (a completion aid). After selecting one of these, entering a “.” (dot or period) displays symbols at the next level. Use the following procedure to enter absolute addresses instead of symbol names:

1. Type the first character of the absolute address.
2. Press the Esc (Escape) key.
3. Type the rest of the absolute address.

Adding a Connection

If you want to configure a connection for an ActiveX control before you place the control into your ActiveX container, you can use the “Add” button to add an instance of the control to the Controls list. Click on the “Add” button to specify the instance that you want to connect to the Data control.

After you have added the ActiveX control instance to the Controls list, you can select the instance from the list, choose the “Add” button again and add any additional properties. For instance, you could add an Edit control instance to the connections list, and then add the Value property to the Edit control in order to assign a variable to the Edit control.

Deleting a Connection

If you remove a control from the ActiveX container, the connection remains configured in the Data control. This means that the next time you place a control of the same name into the container, the connection that you configured for the previous control of that name is automatically applied to the new control. For instance, if you remove a control called Edit1, and later insert a new Edit control, the default name for the new control is Edit1, and the new control inherits the existing Edit1 connection. Use the “Delete” button if you want to prevent new controls from inheriting a previously configured connection: from the Controls list, select the instance whose connection you want to delete, and click on the “Delete” button.

Note

If you remove a control, or change your mind about adding a control after you have already configured a connection for it using the “Add” button, you can only delete the connection to it if there is no control in the ActiveX container that uses the name specified in the connection. Delete the connection before you add any other control that uses the name that is specified in the connection. You cannot use the “Delete” button to remove a connection to a control that is present in the ActiveX container.

5.5 Filtering the Properties for the ActiveX Controls

The Data control provides a filter that allows you to display a subset of the properties for the controls. For example, you may want to display only the Enabled and Value properties and avoid scrolling through all of the other properties for each control.

Use the following procedure for filtering the properties:

1. Access the Properties dialog box for the Data control.
2. As shown in Figure 5-9, click on the “Filter” button.

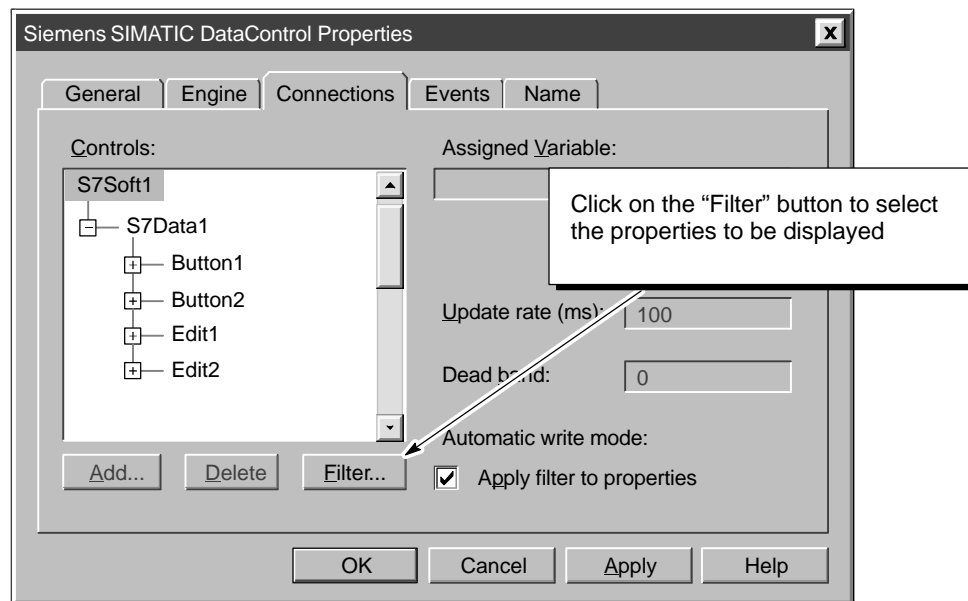


Figure 5-9 Data Control Properties (Connections Tab)

3. As shown in Figure 5-10, enter the properties to display and click on the “Add” button. Use the “Edit” button to correct entries and the “Delete” button to remove entries.
4. As shown in Figure 5-11, select the “Apply filter to properties” check box to display only the properties listed in the filter.

Use the “Apply filter to properties” check box to toggle the filter on and off.

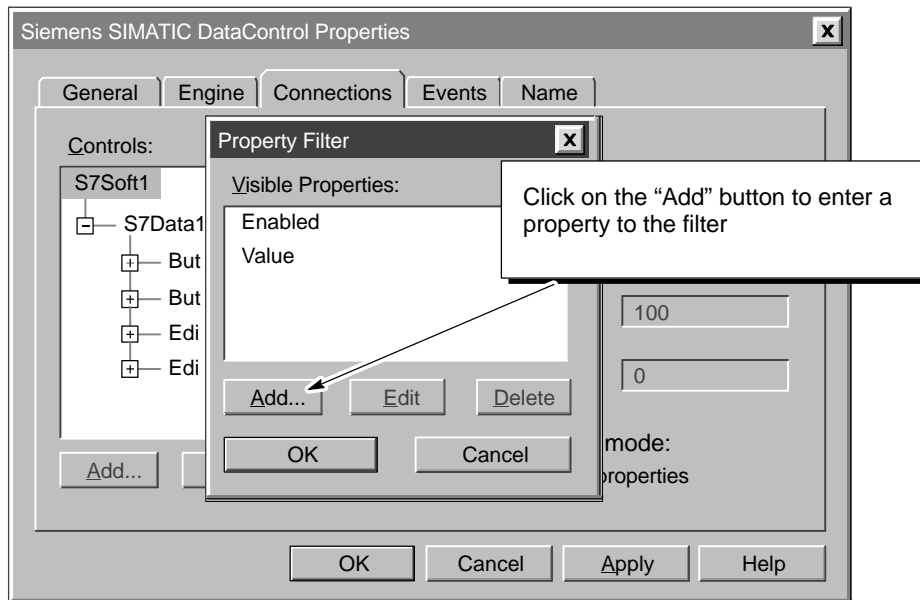


Figure 5-10 Data Control Properties (Connections Tab)

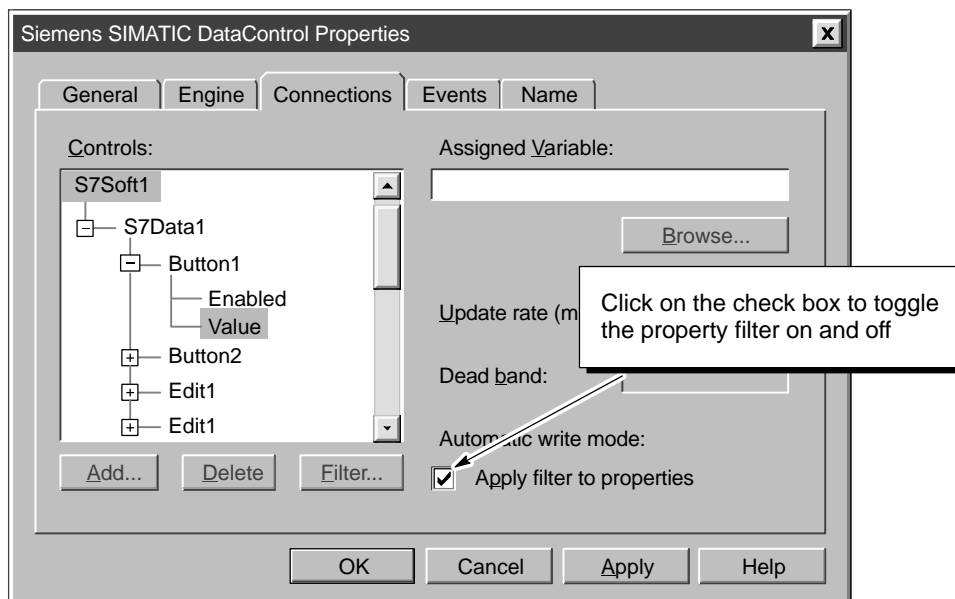


Figure 5-11 Data Control Properties (Connections Tab)

5.6 Configuring Custom Events

As shown in Figure 5-12, the “Events” tab allows you to add custom events that are triggered by the Data control. You enter a key (string) and assign that key to a memory location (variable). If that variable changes, then the Data control generates an event with a parameter that contains the string that you entered in the “Key” field. Your program can then react to this event.

See Section 5.9 for a sample program that responds to events in the CPU.

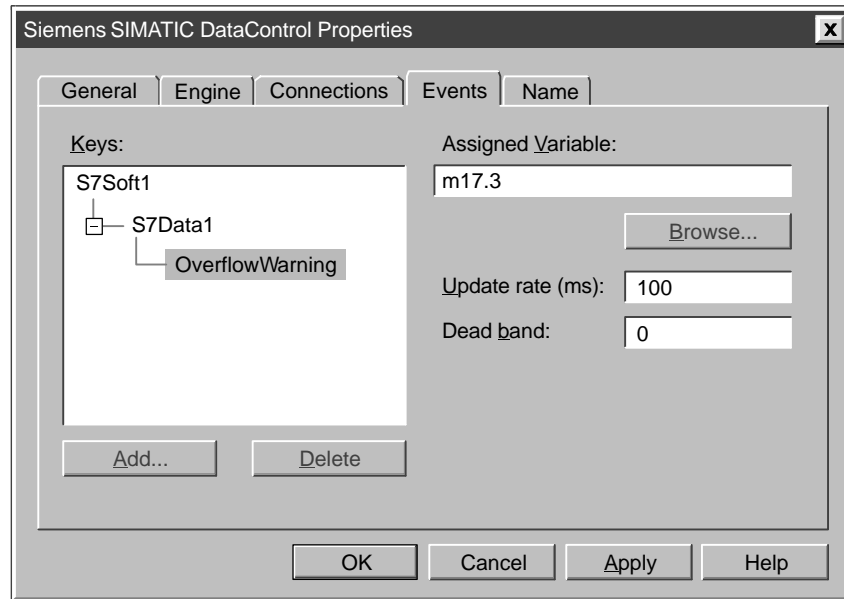


Figure 5-12 Data Control Properties (Events Tab)

Adding an Event

The “Add” button allows you to add user-specified events for controller value changes. You can write your own code to handle the event by handling the ValueChanged event of a Data control. Select a Data control from the expandable list under the “Keys” field and click on the “Add” button; then type any name you choose for the event (for example, OverflowWarning). Next, type a variable in the “Assigned Variable” field to identify the process value that causes the event to be fired.

Figure 5-12 shows an sample event that has been added. A change in the value of M17.3 calls the event handler for the Data control. The input to the event handler is the text string **OverflowWarning**.

Deleting an Event

To delete a user-specified event, expand the list under “Keys,” select the desired event, and click on the “Delete” button.

5.7 Creating a Connection Table

The Data control uses a connection table for determining which properties of the various controls are connected to a specific memory locations of the control engine. Each connection table contains an entry for each connection. Each entry contains the following information:

- Property name: this field identifies the property that has an assigned variable.
- Data source: this field identifies the memory location in the control engine for the connection.
- Update rate: this field defines the update rate for the connection. If no value is entered in this field, the Data control uses the default update rate (which is the value stored in the DefaultUpdateRate property).
- Dead band: this field defines the dead band for automatically writing to the control engine or the control. If no value is entered in this field, the Data control uses the default update rate (which is the value stored in the DefaultDeadBand property).

When you use the “Properties” dialog box to configure the Data control, the Data control automatically creates a connection table. You can also create a program to manually create connection tables. See Section 5.8 for a sample program for manually creating a connection table.

5.8 Sample Program for Creating a Connection Table and an Event Table

You can write a program to create a table to define the connections (assigned variables) or events for the control engine.

Using a Connection Table

You can create a connection table to assign a variable in the control engine to a specific control. The connection table corresponds to the “Connections” tab on the “Properties” dialog box of the Data control.

For each element in the connection table, you must define the property in the control for the connection, the source (memory location of the assigned variable in the control engine), the update rate, and the dead band value. In order to programmatically change connections with a connection table, you must first disconnect the data control (ending all connections) before you can reassign connections and reconnect the Data control.

Note

Instead of creating a connection table, consider using the read and write methods for the Data control (ReadVariable, ReadMultipleVariables, WriteVariable, and WriteMultipleVariables). These methods allow you to access more data with just one line of code.

Table 5-1 shows sample Visual Basic code for a Label control called lblChange in your form to MW2 in the control engine. The value stored in MW2 displays as the caption in the label control.

Table 5-1 Sample Program for Manually Creating a Connection Table

Visual Basic Code	
Dim ControlTable (4) As String	
'Define a connection table for lblChange ControlTable (0) = "Caption" 'Property ControlTable (1) = "MW2:WORD" 'Source (memory location) ControlTable (2) = "100" 'Update rate ControlTable (3) = "0.0" 'Dead band	
'Attach the connection table to S7Data1' S7Data1.ConnectObject lblChange, ControlTable	
'Connect to the control engine S7Data1.Connect 'Connects to the control engine	

Using an Event Table

You can also create an event table to define events for the control engine. The event table corresponds to the “Events” tab of the “Properties” dialog box of the Data control. Table 5-2 provides the sample Visual Basic code for creating an event table.

Section 5.9 provides a sample program that responds to events. This sample program uses an event table to define the events for control engine.

Table 5-2 Sample Program for Manually Creating an Event Table

Visual Basic Code	
Dim controlTable(4) AS String	
'Define the event keys	
ControlTable(0)="M0_0"	'Event Name
ControlTable(1)="M0.0"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_1"	'Event Name
ControlTable(1)="M0.1"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_2"	'Event Name
ControlTable(1)="M0.2"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_3"	'Event Name
ControlTable(1)="M0.3"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_4"	'Event Name
ControlTable(1)="M0.4"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_5"	'Event Name
ControlTable(1)="M0.5"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_6"	'Event Name
ControlTable(1)="M0.6"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_7"	'Event Name
ControlTable(1)="M0.7"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
End	

5.9 Sample Program for Responding to Events

You can create a program that responds to events in the control engine. In this sample program, eight lights correspond to the eight events which are defined in a connection table. (See Figure 5-13.) The events are connected to the status of memory location MB0: a change in the value stored in MB0 generates a set of events (named for each bit in the byte).

As shown in Figure 5-13, the program also contains the following elements:

- Data control (S7Data4) for connecting to the control engine
- Timer (Timer1) that increments the value stored in MB0 (which then causes the control engine to generate the events)
- Command button (cmdStartEvent) for starting or stopping the timer (thereby starting or stopping the generation of events)

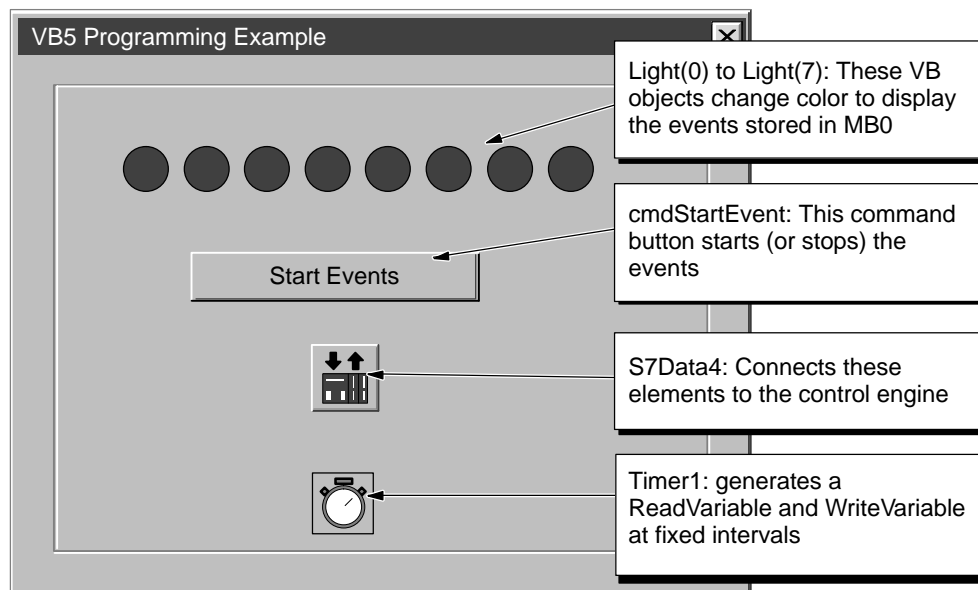


Figure 5-13 Sample Program for Responding to Events in the Control Engine



Caution

Failing to disable the timers in your program could cause timer-generated connections to remain connected, allowing these connections to continue to write data to the control engine. This could cause the control engine to operate erratically, which could potentially cause damage to equipment and injury to personnel.

To ensure that all connections are disconnected when your program closes, always disable all timers before the End statement in the Form_Unload subroutine.

Creating a Connection Table for Responding to Events

Your program can create an event table to define specific events in the control engine. Table 5-3 lists the code for creating a connection table for defining event keys for a control engine.

Table 5-3 Sample Program for Creating a Connection Table for Responding to Events

Visual Basic Code	
Dim controlTable(4) AS String	
'Define the event keys	
ControlTable(0)="M0_0"	'Event Name
ControlTable(1)="M0.0"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_1"	'Event Name
ControlTable(1)="M0.1"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_2"	'Event Name
ControlTable(1)="M0.2"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_3"	'Event Name
ControlTable(1)="M0.3"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_4"	'Event Name
ControlTable(1)="M0.4"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_5"	'Event Name
ControlTable(1)="M0.5"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_6"	'Event Name
ControlTable(1)="M0.6"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
ControlTable(0)="M0_7"	'Event Name
ControlTable(1)="M0.7"	'Process Variable
ControlTable(2)="500"	'Update Rate
ControlTable(3)="0"	'DeadBand
IResult=S7Data1.ConnectName("",ControlTable)	
End	

Responding to the Events Generated by the Sample Program

Table 5-4 provides sample Visual Basic code for responding to different events from the control engine.

Table 5-4 Sample Program for Responding to Events in the Control Engine

Visual Basic Code	
<pre>Private Sub S7Data4_ValueChanged(ByVal Property As String, ByVal VarName As String, ByVal Value As Variant, ByVal Quality As Integer) 'Evaluates which event occurred Select Case Property</pre>	
<pre> Case "M0_0" If Value = True Then Light(0).FillColor = vbGreen Else Light(0).FillColor = vbRed End If</pre>	'Event M0_0 turns Light(0) green
<pre> Case "M0_1" If Value = True Then Light(1).FillColor = vbGreen Else Light(1).FillColor = vbRed End If</pre>	'Event M0_1 turns Light(1) green
<pre> Case "M0_2" If Value = True Then Light(2).FillColor = vbGreen Else Light(2).FillColor = vbRed End If</pre>	'Event M0_2 turns Light(2) green
<pre> Case "M0_3" If Value = True Then Light(3).FillColor = vbGreen Else Light(3).FillColor = vbRed End If</pre>	'Event M0_3 turns Light(3) green
<pre> Case "M0_4" If Value = True Then Light(4).FillColor = vbGreen Else Light(4).FillColor = vbRed End If</pre>	'Event M0_4 turns Light(4) green
<pre> Case "M0_5" If Value = True Then Light(5).FillColor = vbGreen Else Light(5).FillColor = vbRed End If</pre>	'Event M0_5 turns Light(5) green
<pre> Case "M0_6" If Value = True Then Light(6).FillColor = vbGreen Else Light(6).FillColor = vbRed End If</pre>	'Event M0_6 turns Light(6) green

Table 5-4 Sample Program for Responding to Events in the Control Engine, continued

<pre>Case "M0_7" If Value = True Then Light(7).FillColor = vbGreen Else Light(7).FillColor = vbRed End If End Select End Sub</pre>	<pre>'Event M0_7 turns Light(7) green</pre>
--	---

Running the Sample Program (Generating the Events in the Control Engine)

Table 5-5 provides sample Visual Basic code for changing the value stored in MB0. Changing the value of MB0 then causes the control engine to generate the events defined in the connection table (Table 5-3).

- The command button (cmdStartEvents) starts or stops the timer (Timer1).
- The timer (Timer1) reads the value stored in MB0 of the control engine, increments the value, and writes the new value back to the control engine.

The changed value of MB0 causes the control engine to generate the events.



Caution

Failing to disable the timers in your program could cause timer-generated connections to remain connected, allowing these connections to continue to write data to the control engine. This could cause the control engine to operate erratically, which could potentially cause damage to equipment and injury to personnel.

To ensure that all connections are disconnected when your program closes, always disable all timers before the End statement in the Form_Unload subroutine.

Table 5-5 Other Subroutines for Running the Sample Program

Visual Basic Code
<pre>Private Sub cmdStartEvents_Click() If cmdStartEvents.Caption = "Start Events" Then Timer1.Enabled = True cmdStartEvents.Caption = "Stop Events" Else Timer1.Enabled = False cmdStartEvents.Caption = "Start Events" End If End Sub</pre>
<pre>Private Sub Timer1_Timer() Dim mb0 As Variant Dim my_state As Long S7Data4.ReadVariable "MB0", mb0, my_state, 0 If mb0 < 254 Then mb0 = mb0 + 1 Else mb0 = 0 End If Label12.Caption = mb0 S7Data4.WriteVariable "MB0", mb0, 0 End Sub</pre>

5.10 Sample Programs for Reading and Writing Data

You can write a program that initiates access to data (reading or writing) in the control engine.

You can read or write single variables, multiple variables or arrays of variables. (For reading and writing Boolean data, you must use the ReadMultiVariables method or WriteMultiVariables method. See Section 5.11.)

For information about the memory areas of the S7 controllers, see Appendix A.



Caution

Using the timer function improperly or using breakpoints in Visual Basic with Computing can cause problems that could potentially cause your computer or application to crash or lock up. Depending on the configuration, this could cause the application to lose communication with the control engine. This could cause unpredictable process operation, which could result in death or serious injury to personnel, and/or damage to equipment.

Always install a physical emergency stop circuit for your machine or process.

Concerning VB timers: The Timer function in Visual Basic version 5 allows a timer to interrupt code in progress within the same thread, which can cause problems with potentially serious consequences. If you use VB timers with Computing, observe the following guidelines:

- Always kill (disable) the timers in the Form_Unload subroutine. Otherwise, a timer can trigger an event while the VB program is shutting down; this condition could cause your computer or your application to crash, lock up, or to continue running invisibly.
 - If you start your timer in the Form_Load subroutine, the timer event could occur before the other objects have been instantiated. In order to ensure that the objects have been properly instantiated, always start a timer in the Form_Load subroutine with a large interval (such as 1 or 2 seconds) to allow the objects to be properly instantiated. Subsequent timer intervals can be set to shorter intervals.
-

Reading a Single Variable in the Control Engine

Table 5-6 provides sample Visual Basic code for using the ReadVariable method of the Data control to read a single variable in the control engine.

Table 5-6 Reading a Single Variable from the Control Engine

Visual Basic Code
<pre>Private Sub ReadSingleRealVariable Dim rc As Long Dim name_s As String Dim value_v As Variant Dim state_l As Long Dim timeout_l As Long 'Read one Real (floating point) value name_s = "MD0:REAL" timeout_l = 0 rc = S7Data3.ReadVariable(name_s, value_v, state_l, timeout_l) 'Display the value and return code in a List Box ListBox1.Clear ListBox1.AddItem "RetCode = " & Hex(rc) ListBox1.AddItem " - " & name_s & " = " & value_v ListBox1.AddItem " - State = " & Hex(state_l) End Sub</pre>

Writing a Single Variable to the Control Engine

Table 5-7 provides sample Visual Basic code for using the WriteVariable method of the Data control to write a single variable to the control engine.

Table 5-7 Writing a Single Variable to the Control Engine

Visual Basic Code
<pre>Private Sub WriteSingleRealVariable Dim rc As Long Dim name_s As String Dim value_v As Variant Dim timeout_l As Long 'Write one Real (floating point) value name_s = "MD0:REAL" value_v = (Rnd * 1000) timeout_l = 100 rc = S7Data3.WriteVariable(name_s, value_v, timeout_l) 'Display the value and return code in a List Box ListBox1.Clear ListBox1.AddItem "Wrote " & name_s & " = " & value_v ListBox1.AddItem "Return Code = " & Hex(rc) End Sub</pre>

Reading an Array in the Control Engine

Table 5-8 provides sample Visual Basic code for using the ReadVariable method of the Data control to read an array of data in the control engine.

Note

Read and write STRING or CHAR data as Visual Basic BSTR data. Do not use an array of CHAR to emulate a STRING.

Use one BSTR for each STRING or CHAR, regardless of the length of the data being accessed. For example:

- To access CHAR[50] (which designates 50 bytes or 50 characters), use a BSTR of up to 50 bytes and **not** 50 individual BSTRs.
 - To access STRING[50] (which designates a string of 50 characters), use a BSTR of up to 50 bytes and **not** 50 BSTRs.
-

Table 5-8 Sample Program for Reading an Array of Variables

Visual Basic Code
<pre>Private Sub ReadArrayOfReals Dim rc As Long Dim name_s As String Dim value_v As Variant Dim state_1 As Long Dim timeout_1 As Long Randomize 'Read an array of Real (floating point) values name_s = "MD0:Real[3]" timeout_1 = 0 rc = S7Data3.ReadVariable(name_s, value_v, state_1, timeout_1) 'Display the values and return codes for the array in a List Box ListBox1.Clear ListBox1.AddItem "Return Code = " & Hex(rc) ListBox1.AddItem " - name_s & " = " & value_v(0) & " " & value_v(1) & " " & value_v(2) ListBox1.AddItem " - State = " & Hex(state_1) End Sub</pre>

Writing an Array to the Control Engine

Table 5-8 provides sample Visual Basic code for using the WriteVariable method of the Data control to write an array of data to the control engine.

Note

Read and write STRING or CHAR data as Visual Basic BSTR data. Do not use an array of CHAR to emulate a STRING.

Use one BSTR for each STRING or CHAR, regardless of the length of the data being accessed. For example:

- To access CHAR[50] (which designates 50 bytes or 50 characters), use a BSTR of up to 50 bytes and **not** 50 individual BSTRs.
- To access STRING[50] (which designates a string of 50 characters), use a BSTR of up to 50 bytes and **not** 50 BSTRs.

Table 5-9 Sample Program for Writing an Array of Variables

Visual Basic Code
<pre>Private Sub WriteArrayOfReals Dim rc As Long Dim name_s As String Dim timeout_l As Long Dim value_b(2) As Byte ' for byte write Dim value_w(2) As Integer ' for word write Dim value_r(2) As Single ' for real write 'Read an array of Real (floating point) values name_s = "MD0:REAL[3]" value_r(0) = (Rnd * 1000) value_r(1) = (Rnd * 1000) value_r(2) = (Rnd * 1000) timeout_l = 100 rc = S7Data3.WriteVariable(name_s, value_r, timeout_l) 'Display the values and return codes for the array in a List Box ListBox1.Clear ListBox1.AddItem "Return Code = " & Hex(rc) ListBox1.AddItem " - Wrote MD0:REAL[0] = " & value_r(0) ListBox1.AddItem " - Wrote MD0:REAL[1] = " & value_r(1) ListBox1.AddItem " - Wrote MD0:REAL[2] = " & value_r(2) End Sub</pre>

Reading Multiple Variables in the Control Engine

Table 5-10 provides sample Visual Basic code for using the ReadMultiVariables method of the Data control to read multiple variables in the control engine.

Table 5-10 Reading Multiple Variables in the Control Engine

Visual Basic Code
<pre> Private Sub ReadMultiReals Dim i As Integer Dim rc As Long Dim names_array(2) As String Dim values_v As Variant Dim states_v As Variant 'Read three Real (floating point) values For i = 0 To 2 names_array(i) = "MD" & i * 4 & ":REAL" Next i rc = S7Data3.ReadMultiVariables(names_array, values_v, states_v) 'Display the value and return code in a List Box ListBox1.Clear ListBox1.AddItem "RetCode = " & Hex(rc) For i = 0 To 2 ListBox1.AddItem " - " & names_array(i) & " = " & values_v(i) & - vbTab & " State = " & Hex(states_v(i)) Next i End Sub </pre>

Writing Multiple Variables to the Control Engine

Table 5-11 provides sample Visual Basic code for using the WriteMultiVariables method of the Data control to write several variables to the control engine.

Table 5-11 Writing Multiple Variables to the Control Engine

Visual Basic Code
<pre> Private Sub cmdWriteMultVar_Click(Index As Integer) Dim i As Integer Dim rc As Long Dim names_array(2) As String Dim values_v(2) As Variant Dim states_v As Variant 'Write three Real (floating point) values For i = 0 To 2 names_array(i) = "MD" & i * 4 & ":REAL" values_v(i) = (Rnd * 1000) Next i rc = S7Data2.WriteMultiVariables(names_array, values_v, states_v) 'Display the values and return codes in a List Box lstReal.Clear lstReal.AddItem "RetCode = " & Hex(rc) For i = 0 To 2 lstReal.AddItem " - " & names_array(i) & " = " & values_v(i) & vbTab & - " State = " & Hex(states_v(i)) Next i End Sub </pre>

5.11 Sample Program for Reading and Writing Boolean Data

For reading and writing Boolean data, you must use the ReadMultiVariables method or WriteMultiVariables method. Table 5-12 provides a sample program for reading and writing arrays of Boolean data.

Table 5-12 Reading and Writing Multiple Variables

Visual Basic Code
<pre> Private Sub Read_Booleans() Dim mybooleans(7) As String Dim vals_v As Variant Dim states_v As Variant Dim rc As Long mybooleans(0) = "m0.0" mybooleans(1) = "m0.1" mybooleans(2) = "m0.2" mybooleans(3) = "m0.3" mybooleans(4) = "m0.4" mybooleans(5) = "m0.5" mybooleans(6) = "m0.6" mybooleans(7) = "m0.7" rc = S7Data1.ReadMultiVariables(mybooleans, vals_v, states_v) End Sub </pre>
<pre> Private Sub Write_Booleans() Dim mybooleans(7) As String Dim myvals(7) As Variant Dim states_v As Variant Dim rc As Long mybooleans(0) = "m0.0" mybooleans(1) = "m0.1" mybooleans(2) = "m0.2" mybooleans(3) = "m0.3" mybooleans(4) = "m0.4" mybooleans(5) = "m0.5" mybooleans(6) = "m0.6" mybooleans(7) = "m0.7" myvals(0) = False myvals(1) = False myvals(2) = False myvals(3) = False myvals(4) = False myvals(5) = False myvals(6) = False myvals(7) = False rc = S7Data1.WriteMultiVariables(mybooleans, myvals, states_v) End Sub </pre>

5.12 Properties, Methods, and Events of the Data Control

You use the properties and methods listed in Table 5-13 to manipulate the Data control.

Table 5-13 Properties and Methods of the Data Control

Property or Method	Description	Page
Activated property	Specifies whether or not all connections are activated	B-1
AutoConnect property	Specifies whether or not the configured connections are established at runtime	B-3
AutoConnectTimeout property	Specifies a timeout value	B-4
Connect method	Establishes all configured connections	B-7
ConnectName method	Establishes connections for the object that is specified by name	B-8
ConnectObject method	Establishes connections for a specified object	B-9
ControlEngine property	Specifies the control engine for the connection	
DefaultDeadband property	Specifies the dead band used by the Data control, if no dead band is specified in the connection table	B-12
DefaultUpdateRate property	Specifies the update rate used by the Data control, if no update rate is specified in the connection table	B-13
Disconnect method	Releases all established connections	B-14
MultipleEngines property	Specifies whether the connection is to one specific control or to several control engines	B-27
PCName property	Specifies the network identification for a remote computer (for connecting over a network)	B-28
PropertyChangedName method	Notifies the Data control that the value of a property of a connected control, referenced by Name, has changed	B-30
PropertyChangedObject method	Notifies the Data control that the value of a property of a connected control, referenced by Object, has changed	B-31
ReadMultiVariables method	Reads the status of several variables in the control engine	B-33
ReadVariable method	Reads the status of one specific variable in the control engine	B-33
ShowErrorBoxes property	Specifies whether to display the default error boxes when there is a user-generated error	B-36
TagSource property	Specifies the source (such as a tag file) of symbolic information to be used when assigning variables and identifying control engines	B-39
WriteMultiVariables method	Writes new values to several variables in the control engine	B-44
WriteVariable method	Writes a new value to a specific variable in the control engine	B-45

The Data control responds to the events listed in Table 5-14.

Table 5-14 Events of the Data Control

Event	Description	Page
ConnectionError	Occurs when an error on a connection occurs	C-1
ValueChanged	Occurs when the value of a connected variable changes and no connected event was specified on the call to the Connect method	C-9

Error Codes for the Data Control (ConnectionError Event)

When an error occurs in the Data control, the control generates a ConnectionError event. Your program can capture this ConnectionError event and respond to specific situations. The ConnectionError event can detect standard OLE errors, such as E_FAIL or E_OUTOFMEM. Table 5-15 lists some of the error codes.

Table 5-15 Data Control Error Codes

Error Code	Description
0x80004005	General OLE failure
0x8007000E	Out of available memory
0x80070057	Invalid variable syntax
0xC0040004	Invalid or unknown data type
0xC0040007	Invalid variable type
0xC0040008	Invalid syntax for the item definition
0xC004000B	Value passed to WRITE is out of range

User Controls

Chapter Overview

Computing provides ActiveX User controls for accessing process data. You use the Properties dialog box of the Data control to establish a connection between the user control and the control engine. Each control has a properties dialog box for defining the performance of the control.

- Button control allows you to turn individual bits of memory on and off.
- Edit control provides access to the memory locations of the control engine.
- Label control allows you to display a constant string.
- Slider control provides an interface for monitoring and modifying analog variables.

Section	Description	Page
6.1	Connecting the User Controls to the Process Data	6-2
6.2	Using the Property Pages of the Button Control	6-4
6.3	Properties and Methods of the Button Control	6-9
6.4	Events of the Button Control	6-10
6.5	Using the Property Pages of the Edit Control	6-11
6.6	Properties and Methods of the Edit Control	6-18
6.7	Events of the Edit Control	6-19
6.8	Error Codes for the Edit Control	6-20
6-21	Using the Property Pages of the Label Control	6-21
6.10	Properties and Methods of the Label Control	6-26
6.11	Events of the Label Control	6-26
6.12	Using the Property Pages of the Slider Control	6-27
6.13	Properties and Methods of the Slider Control	6-34
6.14	Events of the Slider Control	6-35

6.1 Connecting the User Controls to the Process Data

To establish a connection between a Button, Edit, or Slider control and your process data, you assign a (single-bit) variable to the Value property of the control. To establish a connection between the Label control and your process data, you assign a (single-bit) variable to the Caption property of the Label control. The variable cannot be assigned within the Properties dialog box of the control. Instead, use the Properties dialog box of the Data control and select the button from the expandable list of controls under the Connections tab. See Figure 6-1.

To set properties for anything other than the Value property, you can use the Properties dialog box of the control itself. Use the **Edit** menu or right click the mouse button and select the **Properties** command for the control.

Note

In order to connect the control to actual process data, you must establish a connection through the Data control.

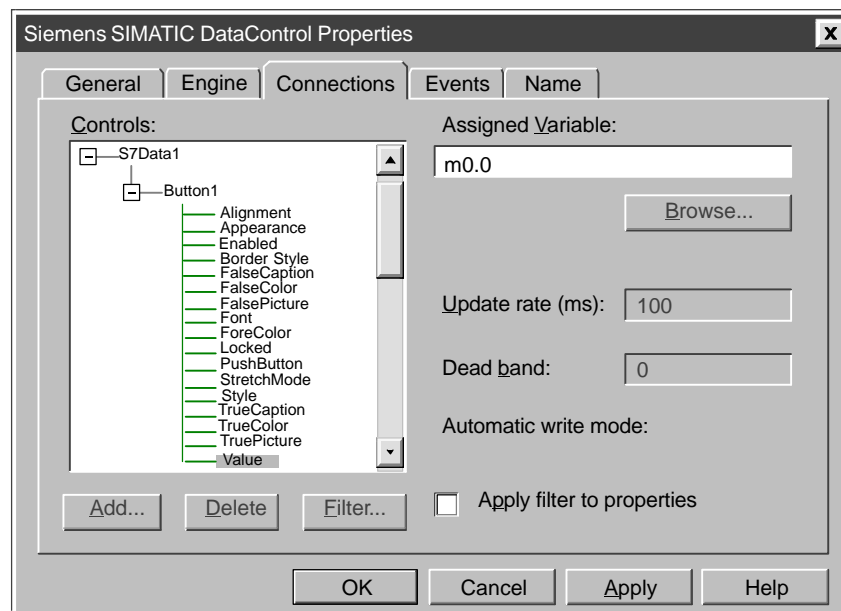


Figure 6-1 Assigning Variables for a Button, Edit, Control

Specifying Variables and Data Types

While Computing allows you to specify a data type when you assign a variable to one of the properties of an S7 control, remember that the Button control can be assigned only to an individual bit in the control engine. The only valid data type for a Button control is BOOL.

As an option, Computing allows you to specify a data type when you assign a variable to one of the properties of a SIMATIC control. You define the data type by entering the absolute address for the memory location, followed by a colon (:) and then the data type. Be careful in assigning data types. If you are connecting an Edit control, the values for some S7 datatypes will not display properly unless the datatype you assign matches the Data Format field in the Edit Control properties box. For example, you can define an assigned variable as a REAL data type by entering “MD100:real” when you assign the variable, but you must be sure to set the Data Format field in the Edit Control properties box to Real.

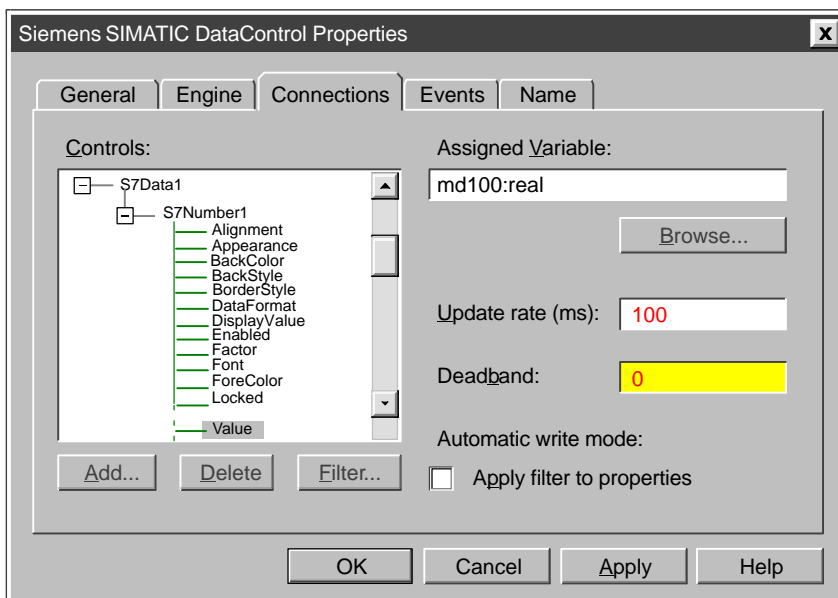


Figure 6-2 Assigning Variables for the Edit Control

6.2 Using the Property Pages of the Button Control

The Button control allows you to associate a button display with a data bit from your process. You associate the button with your process by assigning a variable (namely, the desired bit location) to it. You can then toggle the button display to change the state of the bit; the button color also changes automatically as the state of the bit changes within the process.

The Button control provides access to individual memory bits in the control engine and has two states of animation: 0 (off) or 1 (on). Clicking on the Button control changes the data in the control engine.

The Button control reads and writes Boolean (single bit) values.

Defining the Caption and Enabling the Control (Using the General Tab)

The General tab of the Properties dialog (see Figure 6-3) allows you to define the two captions for the Button control:

- “Alignment” determines the alignment of the text (left, center, or right).
- “TrueCaption”: Enter the text to be displayed in the control when the bit is “true” (equal to 1 or “on”).
- “FalseCaption”: Enter the text to be displayed in the control when the bit is “false” (equal to 0 or “off”).
- “Style” determines the style (standard or graphical) for the control. Graphical style means that a bitmap is used.
- “Appearance”: If you set this property to 3D, the control will have a three-dimensional appearance. (You must also set the border style to Fixed Single to enable the three-dimensional appearance.) The other option is Flat, which displays a two-dimensional, rectangular border around the control.
- “BorderStyle”: If you set this property to Fixed Single, the control is displayed with a rectangular border; if you set the property to None, no border will be displayed.
- “StretchMode” determines the stretch mode of the graphical element for the control.
- “Enabled” checkbox determines whether the Button responds to events. It does not generate events while disabled. The default setting for this option is enabled (selected).
- “Locked” checkbox determines whether the control is in a read only state. In locked state, you cannot change any values.
- “Pushbutton” determines whether or not the control functions like a pushbutton. It determines the operation mode of the control: if set to “True” or 1, the True value is retained as long as the Button control is “pressed” (mouse down).

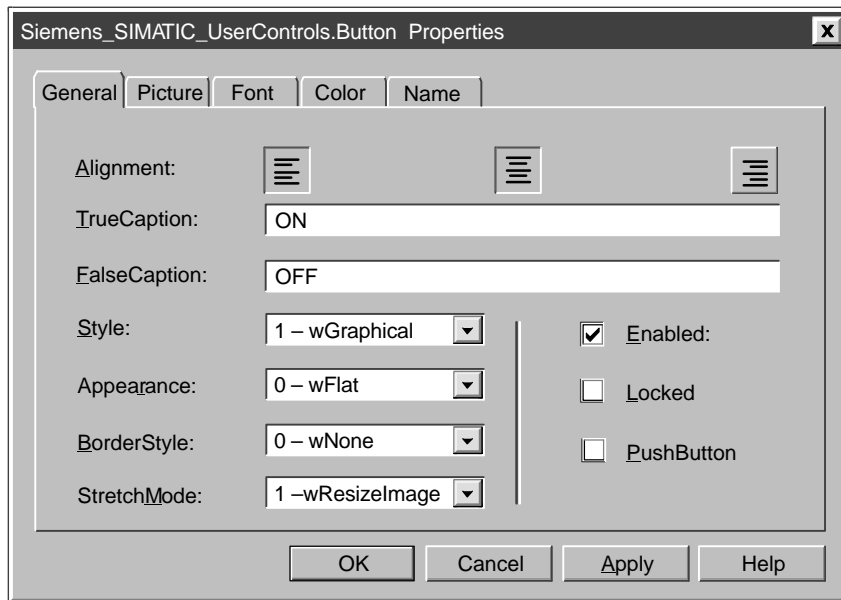


Figure 6-3 Button Control Properties (General Tab)

Defining the Picture of the Button Control (Using the Picture Tab)

The Picture tab of the Properties dialog (see Figure 6-4) allows you to browse to a picture for the two states of the Button control. Select the “Off” state (FalsePicture) or the “On” state (TruePicture) and then click on Browse to select the picture to be displayed for that state. You can use any pictures for the On and Off states, but graphics are only allowed if “1 – wGraphical” is selected in the Style field of the General tab. Prefined bitmaps are provided in WinAC\WinCP\bitmaps.

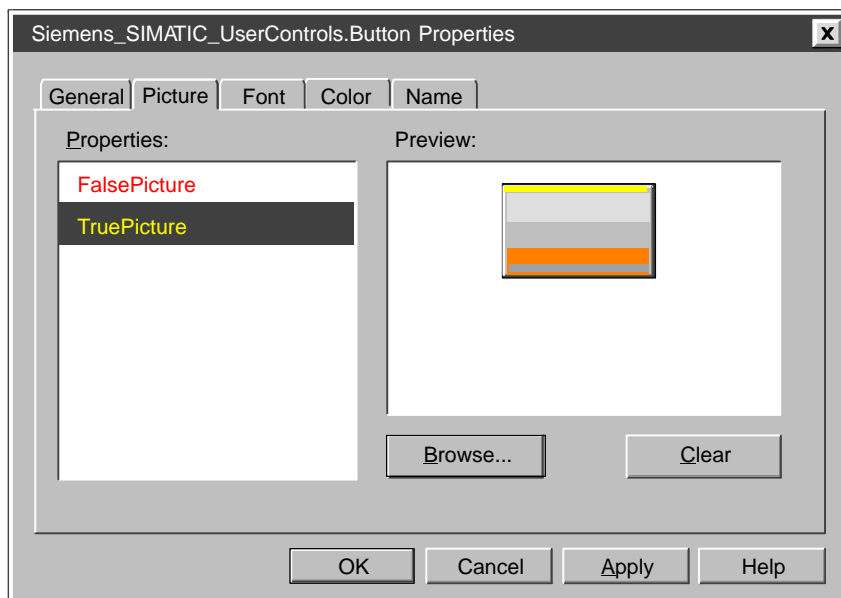


Figure 6-4 Button Control Properties (Picture Tab)

Defining the Typeface of the Button Control (Using the Font Tab)

The Font tab of the Properties dialog (see Figure 6-5) allows you to define the typeface and size for the text on the Button control:

- “Font”: select the typeface for the text from a list of standard typefaces.
- “Size”: select the point size or enter a specific point size for the text.
- “Effects”: select other typographical options (boldface, italic, underline, or strike-through).

The “Sample Text” field displays the selection of the Font property.

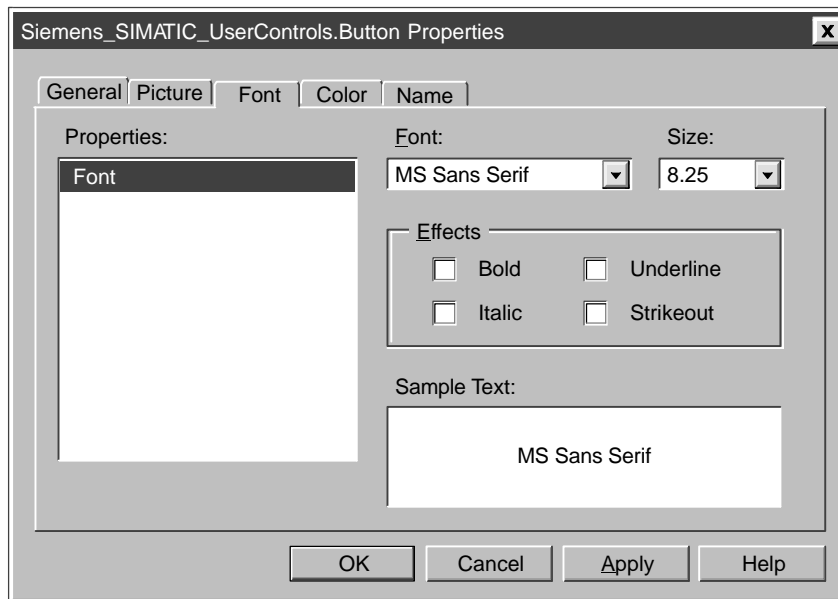


Figure 6-5 Button Control Properties (Font Tab)

Defining the Color of the Button Control (Using the Color Tab)

The Color tab of the Properties dialog (see Figure 6-6) allows you to define the colors for the two states, and for the text of the Button control. You can choose from a palette of standard colors, or you can create custom colors.

- You select the “Off” state (FalseColor) or the “On” state (TrueColor) and then select the color to be displayed for that state from the color palette.
- You can also define the ForeColor which is the color used to display text in an object.

Note

FalseColor and TrueColor can be changed only if you have selected Style: Standard on the General tab, but ForeColor (Text color) can be changed for both Style: Standard and Style: Graphical.

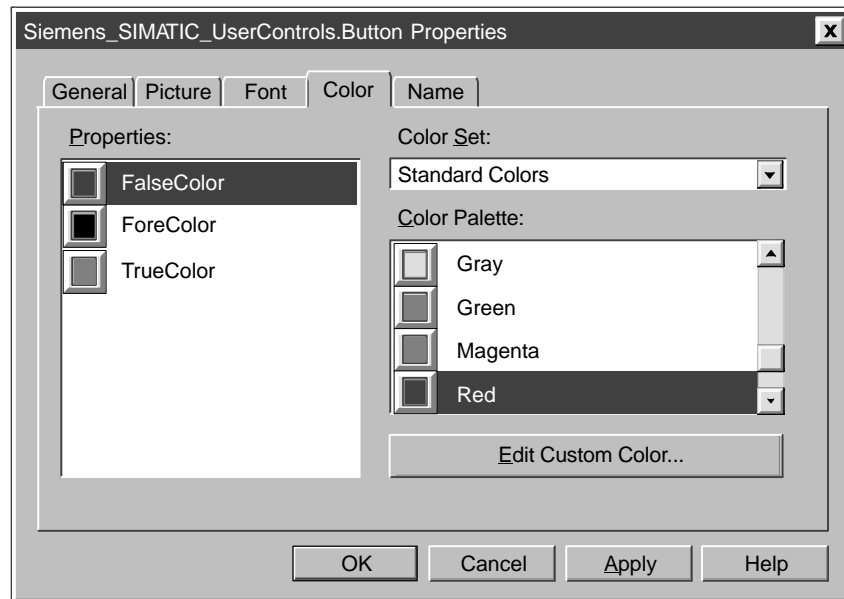


Figure 6-6 Button Control Properties (Color Tab)

Using the Name Tab

The Name tab of the Properties dialog (see Figure 6-12) allows you to assign more meaningful names to the control you have inserted into the container. This tab appears only when you are using controls in the Computing Container.

You type the new name in the “Control Name” field, and click on “Apply” or “OK”. The new name appears in the “Select Control” list in the toolbar. To open the control Properties dialog box, double-click on the desired control.

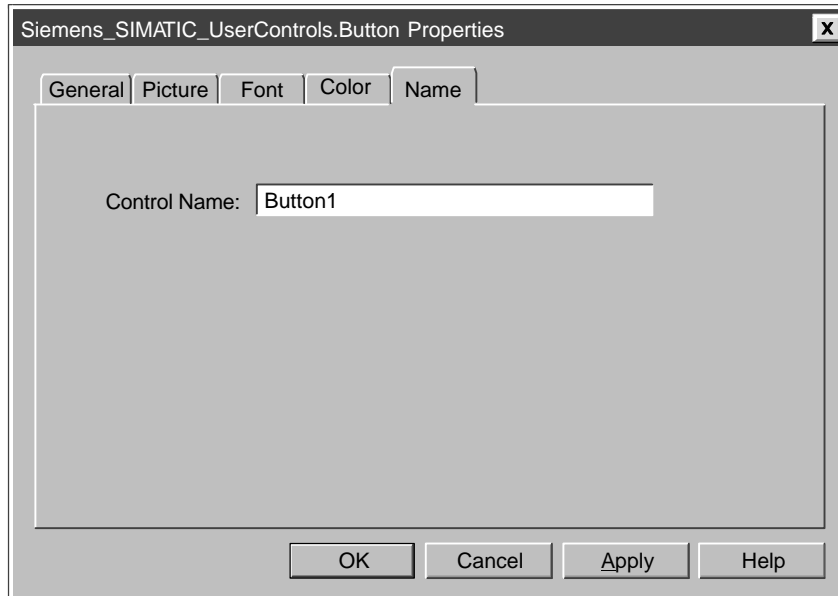


Figure 6-7 Button Control Properties (Name Tab)

6.3 Properties and Methods of the Button Control

You use the properties and methods listed in Table 6-1 to manipulate the Button control.

Table 6-1 Properties and Methods of the Button Control

Property or Method	Description	Page
AboutBox method	Displays the “About” message box for the control	B-1
Alignment property	Determines the alignment of the text	B-2
Appearance property	Determines if the control is displayed with 3D effects	B-3
BorderStyle property	Selects the border style (fixed single, or none)	B-6
Enabled property	Determines whether the control responds to user-generated events	B-19
FalseCaption property	Determines the text that is displayed in the control when the Value property is False (equal to 0, or “Off”)	B-21
FalseColor property	Determines the color of the control when the Value property is False (equal to 0, or “Off”)	B-21
FalsePicture property	Determines the graphic element that is displayed by the control when the Value property is False (equal to 0, or “Off”)	B-22
Font property	Returns a Font object for the main font of the control	B-22
ForeColor property	Determines the foreground color used to display the text of the control	B-23
Locked property	Sets the control to a read-only state. By default, the control is not in locked mode, so the user can enter numbers.	B-26
PushButton property	Determines the operation mode of the control: if set to “True” or 1, the Value property is inverted as long as the Button control is “pressed” (MouseDown event)	B-32
StretchMode property	Determines the “stretch mode” of the graphic element for the control	B-38
Style property	Determines the style (standard or graphical) for the control	B-39
TrueCaption property	Determines the text that is displayed in the control when the Value property is True (equal to 1, or “On”)	B-40
TrueColor property	Determines the color of the control when the Value property is True (equal to 1, or “On”)	B-41
TruePicture property	Determines the graphic element that is displayed by the control when the Value property is True (equal to 1, or “On”)	B-42
Value property	Contains the value that is linked to the control engine	B-43

6.4 Events of the Button Control

The control responds to the events listed in Table 6-2.

Table 6-2 Events of the Button Control

Event	Description	Page
Change event	Occurs when the value of the Value property changes	C-1
Click event	Occurs when a mouse button is pressed and released while the mouse cursor is over the control	C-1
Error event	Occurs when a property is set to an illegal value	C-2
KeyDown event	Occurs when the user presses a key while the control has the focus	C-3
KeyPress event	Occurs when an ANSI key is pressed and released while the control has the focus	C-4
KeyUp event	Occurs when a key is released while the control has the focus	C-5
MouseDown event	Occurs when a mouse button is pressed while the mouse cursor is over the control	C-6
MouseMove event	Occurs when the mouse cursor moves over the control	C-7
MouseUp event	Occurs when a mouse button is released while the mouse cursor is over the control	C-8

Note

In order to connect the Edit control to actual process data, you must establish a connection through the Data control.

6.5 Using the Property Pages of the Edit Control

The Edit control allows you to display process data in a numeric format and to modify that data. You associate the number display with your process by assigning a variable (the process value) to it. You can type a new value into the display; the display also updates automatically when the variable associated with it changes within the process. The Edit control provides access to the memory locations of the control engine. Entering a new value in the control changes the data in the control engine.

Note

Computing does not allow you to write to timers.

Defining How the Data is Displayed (Using the General Tab)

The fields on the General tab allow you to define the following properties concerning how the data will be displayed:

- “Alignment” defines how the value will be displayed in the Edit control: aligned to the left side of the field, centered in the field, or aligned to the right side of the field.
- “Data format” defines the storage type used for converted values. If you are using a data type for displaying a value which is too large, the value will be truncated.

The data type specified in this field must match any data type specified in the “Assigned Variable” field of the S7Data Control Properties dialog box (see Figure LEERER MERKER). Table 6-3 shows the data type sizes for the Edit control.

- “Precision” defines the decimal place for the real (floating-point) number. You enter the number of digits to the right of the decimal. (The default value is three digits.) This field is enabled only for the Real data type.
- “Appearance” defines the way the control looks. If you set this property to 3D, the control will have a three-dimensional appearance. (You must also set the border style to Fixed Single to enable the three-dimensional appearance.) The other option is Flat, which displays a two-dimensional, rectangular border around the control.
- “Border Style” defines whether a border is displayed. If you set this property to Fixed Single, the control is displayed with a rectangular border; if you set the property to None, no border will be displayed.
- “WriteMode” determines how the control responds when the user enters a new value. If the write mode is set to Automatic (0), the value (if valid) is written automatically into the Value property (and to the control engine). If the write mode is Manual (1), the value is not written to the value property unless your program code calls the method “Write” at the control.

Using the check boxes on the General tab, you can define other operations for the control:

- “Enabled” checkbox determines whether the control responds to events. It does not generate events while disabled. The default setting for this option is enabled (selected).
- “Locked”: When you enable this option, the control becomes a read-only display: you can view the value in the memory location of the control engine, but you cannot change the values from this control. The default setting for this option is disabled (not selected).
- “Zero Pad”: When you enable this option, the Edit control fills out the data type by inserting zeroes (0) to the left of the value. The default setting for this option is disabled (not selected).

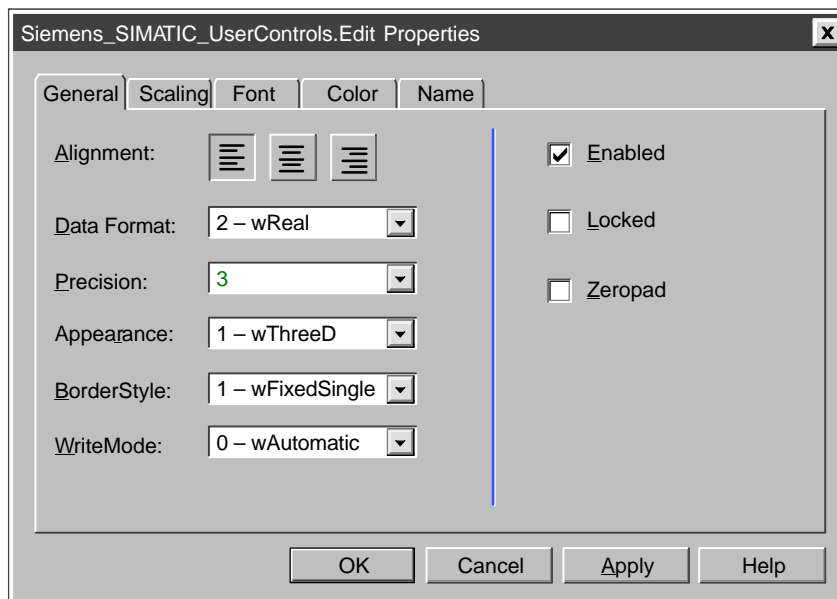


Figure 6-8 Edit Control Properties (General Tab)

Table 6-3 Size of Data Types for the Edit control

Data Type	Setting	Size	Description
Boolean	0	1 bit	Single bit value
Byte	1	1 byte	Unsigned single-byte value
Word	2	1 byte	Unsigned two-byte value
Integer	3	2 bytes	Signed two-byte integer value
Double Word	4	4 bytes	Unsigned four-byte value (default)
Double Integer	5	4 bytes	Signed four-byte integer value
Real	6	4 bytes	Signed four-byte real (floating-point) value
Timer	7	2 bytes	Unsigned two-byte value
Counter	8	2 bytes	Unsigned two-byte value

Using the Scaling Tab

The Scaling tab of the Properties dialog box (see Figure 6-9) allows you to define a scale for displaying the value in the memory location. This scaling factor is used both in reading a value from and writing a value to the control engine. You can select one of three scaling options:

- No scaling of the data (default) (0–wNoScaling)
- Scaling by formula (1–wByFormula)
- Scaling by range (2–wByRange)

No scaling of the data: If you choose the default, the Display Value shows a Max of 100 and a Min of 0.

Scaling by formula: If you choose to scale by formula, you enter the following information:

- Factor represents a percentage of change (scaling factor) from the value in the control engine to the value in the Edit control.
- Offset represents a fixed value to be added to the scaled result before being displayed.

The Edit control uses the following formula to calculate the scaled value:

$$(\text{Value} \times \text{Factor}) + \text{Offset} = \text{Display Value}$$

where:

Value = the value stored in the control engine

Factor = the scaling factor

Offset = the offset factor

Display Value = the value displayed in the Edit control

When the Edit control writes data to the control engine, the inverse of the formula is used to scale the value.

Scaling by range: If you choose to scale by range transformation, you specify the upper (RawMax) and lower (RawMin) values for a source range (for the value in the control engine) and for a destination range (for the value displayed in the Edit control or Display). The Edit control then transforms the value from one range into the equivalent value for the other range.

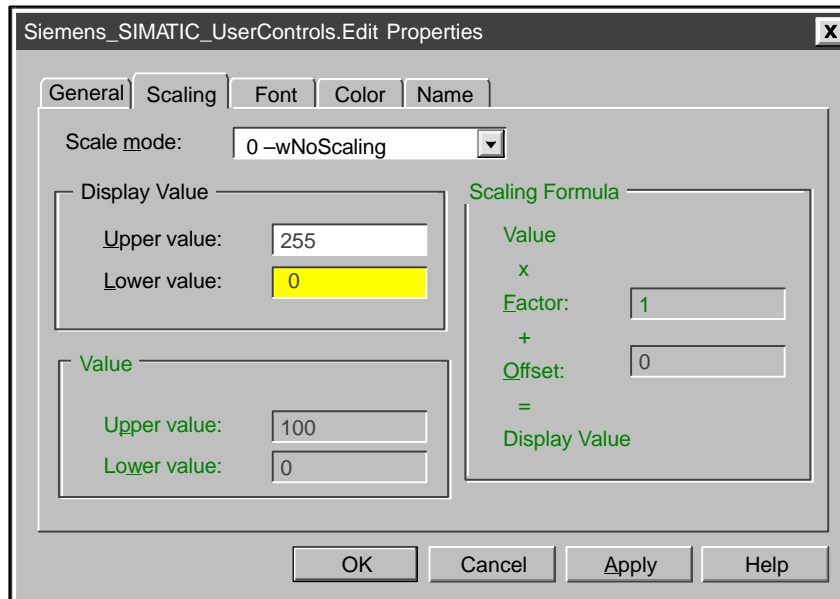


Figure 6-9 Edit Control Properties (Scaling Tab)

These ranges define only the relationship between the data in the control engine and the data in the Edit control: if the value is above or below the ranges entered for the transformation, the transformation uses the formula to extrapolate the scaled value. The upper and lower limits are not “minimum and maximum values” for the data: there is no “limit checking” with the scaling factors.

Defining the Typeface of the Text (Using the Font Tab)

The Font tab of the Properties dialog (see Figure 6-10) allows you to define the typeface and size for the text of the Edit control:

- “Font” selects the typeface for the text from a list of standard typefaces.
- “Size” selects the point size or enter a specific point size for the text.
- “Effects” selects other typographical options (boldface, italic, underline, or strike-through) for the text.

The “Sample Text” field displays the selection of the Font property.

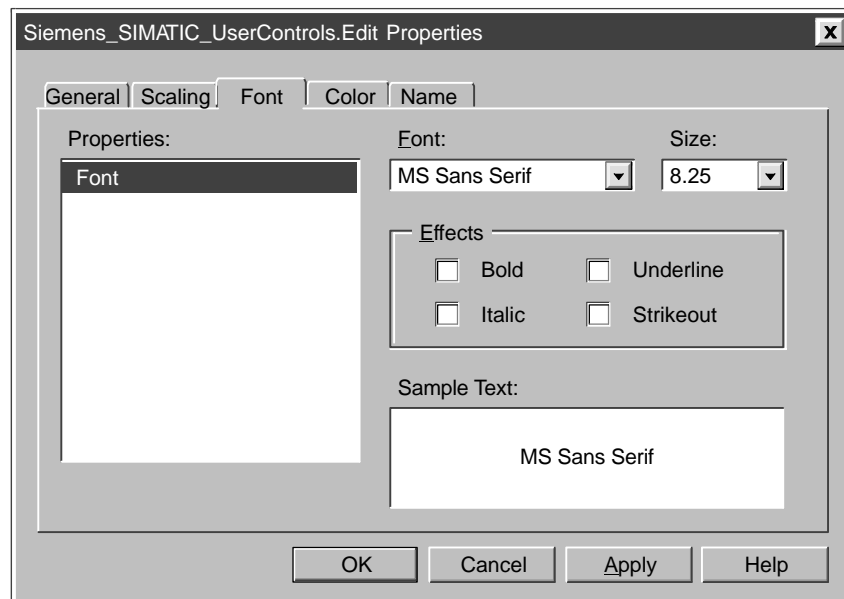


Figure 6-10 Edit Control Properties (Font Tab)

Defining the Color of the Edit Control (Using the Color Tab)

The Color tab of the Properties dialog (see Figure 6-11) allows you to define the colors for the two states, and for the text of the Edit control. You select the Property ("Back Color" or "ForeColor") and then select the color to be displayed for that property from the color palette. You can choose from a palette of standard colors, or you can create custom colors.

Note

BackColor and ForeColor can be changed only if you have selected Style: Standard on the General tab, but ForeColor (Text color) can be changed for both Style: Standard and Style: Graphical.

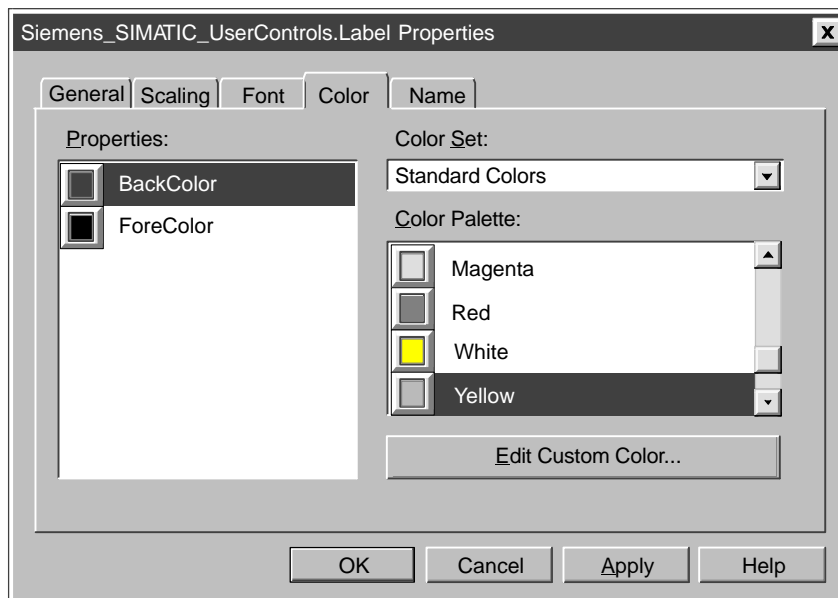


Figure 6-11 Edit Control Properties (Color Tab)

Using the Name Tab

The Name tab of the Properties dialog (see Figure 6-12) allows you to assign more meaningful names to the control you have inserted into the container. This tab appears only when you are using controls in the Computing Container.

You type the new name in the “Control Name” field, and click on “Apply” or “OK”. The new name appears in the “Select Control” list in the toolbar. To open the control Properties dialog box, double-click on the desired control.

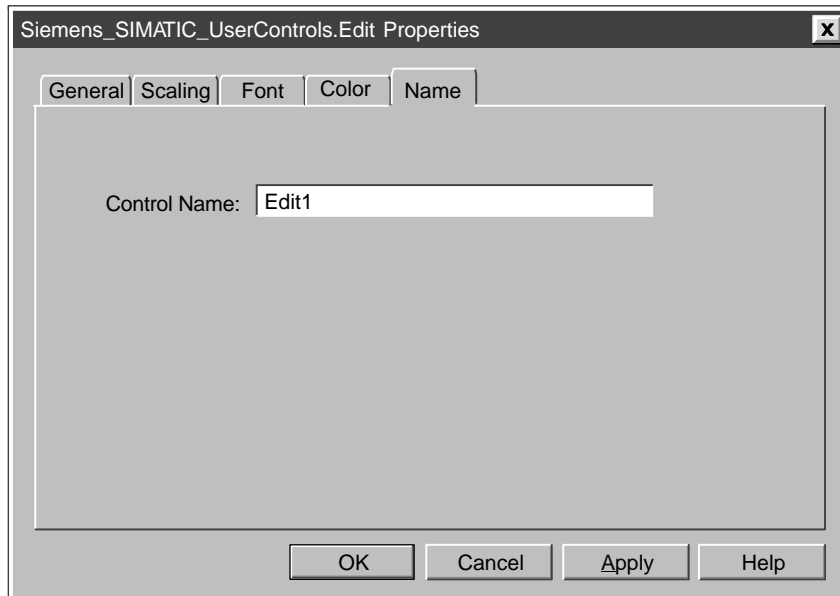


Figure 6-12 Edit Control Properties (Name Tab)

6.6 Properties and Methods of the Edit Control

You use the properties and methods listed in Table 6-4 to manipulate the Edit control.

Table 6-4 Properties and Methods of the Edit Control

Property or Method	Description	Page
AboutBox method	Displays the “About” message box for the control	B-1
Alignment property	Specifies the alignment of the number in the control	B-2
Appearance property	Specifies whether the control is displayed as 3D or flat	B-3
BackColor property	Returns or sets the background color	B-5
BorderStyle property	Selects the border style (fixed single, or none)	B-6
DataFormat property	Defines the storage type used for converted values	B-11
DisplayValue property	Returns the scaled value for the control	B-18
Enabled property	Determines whether the control reacts to changes of the Value property and fires events	B-19
Factor property	Specifies the scaling factor used when the scale-by-formula option has been enabled (Used with ScaleMode property)	B-20
Font property	Returns a Font object for the main font of the control	B-22
ForeColor property	Returns or sets the foreground color used to display text and graphics	B-23
Locked property	Sets the control to a read-only state. By default, the control is not in locked mode, so the user can enter numbers.	B-26
Max property	Returns/sets the maximum scaled value of the control if the ScaleMode property is wByRange or wScaleNone	B-26
Min property	Returns/sets the minimum scaled value of the control if the ScaleMode property is wByRange or wScaleNone	B-26
Offset property	Specifies the offset used when the scale-by-formula option has been enabled (Used with ScaleMode property)	B-27
Precision property	Selects the precision of the Real number	B-30
RawMax property	Defines the upper value of the source range for scaling a value. The ScaleMode property must be set to “wByRange”.	B-32
RawMin property	Defines the lower value of the source range for scaling a value. The ScaleMode property must be set to “wByRange”.	B-32
ScaleMode property	Specifies the scaling mode to be used for scaling the values	B-34
Value property	Contains the value that is linked to the control engine	B-43

Table 6-4 Properties and Methods of the Edit Control, continued

Property or Method	Description	Page
WriteMode property	Selects whether to write new values automatically or manually	B-43
WriteNow method	Writes the value of the Value property	B-44
ZeroPad	Determines whether the displayed number is padded with zeroes (to the left of the value) to the size of the data type	B-45

6.7 Events of the Edit Control

The Edit control responds to the events listed in Table 6-5.

Table 6-5 Events of the Edit Control

Event	Description	Page
Change event	Occurs when the value of the Value property changes	C-1
Click event	Occurs when a mouse button is pressed and released while the mouse cursor is over the control	C-1
DbClick event	Occurs when a mouse button is double-clicked while the cursor is over the control	C-2
Error event	Occurs when a property is set to an illegal value	C-2
KeyDown event	Occurs when the user presses a key while the control has the focus	C-3
KeyPress event	Occurs when an ANSI key is pressed and released while the control has the focus	C-4
KeyUp event	Occurs when a key is released while the control has the focus	C-5
MouseDown event	Occurs when a mouse button is pressed while the mouse cursor is over the control	C-6
MouseMove event	Occurs when the mouse cursor moves over the control	C-7
MouseUp event	Occurs when a mouse button is released while the mouse cursor is over the control	C-8

6.8 Error Codes for the Edit Control

When an error occurs in the Edit control, the control generates an Error event. Your program can capture this Error event and respond to specific situations. Table 6-6 lists the error codes for the Edit control.

Table 6-6 Error Codes for the Edit Control

Error Code	Description
C0040002	<p>The scaling cannot proceed because of an error in the formula used.</p> <p>This error only appears if you are using the Edit control with range scaling. In this case it is possible that you have specified a raw value range (RawMin)(RawMax) of the length of zero (<i>min</i> equal to <i>max</i>). This would lead to a division by zero, which means the scaling is impossible.</p> <p>To correct the error, specify a raw value range where RawMin is not equal to RawMax.</p>
C0040003	<p>The set value at the Value property is invalid.</p> <p>The value which came from the control engine or from a script that is accessing the Value property is not interpretable.</p> <p>To correct the error, check the values that you have written to the control.</p>
C0040004	<p>The set value at the Text property is invalid.</p> <p>This is a common error, which occurs if the user enters an incorrect value in the control. Normally, it means that the entered text contains characters that are not allowed.</p> <p>The allowed characters are dependent on the DataType used.</p> <p>To correct the error, reenter a value that is allowed.</p>
C0040005	<p>The other OLE components could not be found.</p> <p>An error occurred in the installation of Computing or of Windows itself. The control is unable to access the other necessary parts that are needed for the software to work properly.</p> <p>To correct the error, check the installation.</p>
C0040006	<p>The Microsoft standard controls could not be created.</p> <p>Something went wrong with the installation of Computing or Windows itself. The control is unable to access the other necessary parts that are needed for the software to work properly.</p> <p>To correct the error, check the installation.</p>
C0040010	<p>The limit check cannot proceed, because the RawMin is greater than the RawMax.</p> <p>This error can only appear if you are using the Edit control with limit checking (checking for upper and lower limit). In this case it is possible that you've specified a lower limit (RawMin) that is greater than the upper limit (RawMa).</p> <p>To correct the error, specify a valid range for limit checking. The lower limit has to be less than the upper limit.</p>

6.9 Using the Property Pages of the Label Control

The Label control allows you to display a constant string. It is also possible to connect the Caption property of the Label control with any process value. The process value is converted into a string and displayed. The Label control cannot be used for input.

Defining the Label and Enabling the Control (Using the General Tab)

The General tab of the Properties dialog (see Figure 6-13) allows you to define the presentation of the Label control.

- “Alignment” defines how the value will be displayed in the Label control: aligned to the left side of the field, centered in the field, or aligned to the right side of the field.
- “Caption” specifies the text that is displayed by the control. If the caption is attached to a process value, the process value is displayed instead.
- “Style” determines the style (standard or graphical) for the control.
- “Appearance” defines the way the control looks. If you set this property to 3D, the control will have a three-dimensional appearance. (You must also set the border style to Fixed Single to enable the three-dimensional appearance.) The other option is Flat, which displays a two-dimensional, rectangular border around the control.
- “Border Style” defines whether a border is displayed. If you set this property to Fixed Single, the control is displayed with a rectangular border; if you set the property to None, no border will be displayed.
- “StretchMode” determines the stretch mode of the graphical element for the control.
- “Enabled” checkbox determines whether the Label responds to events. It does not generate events while disabled. The default setting for this option is enabled (selected).

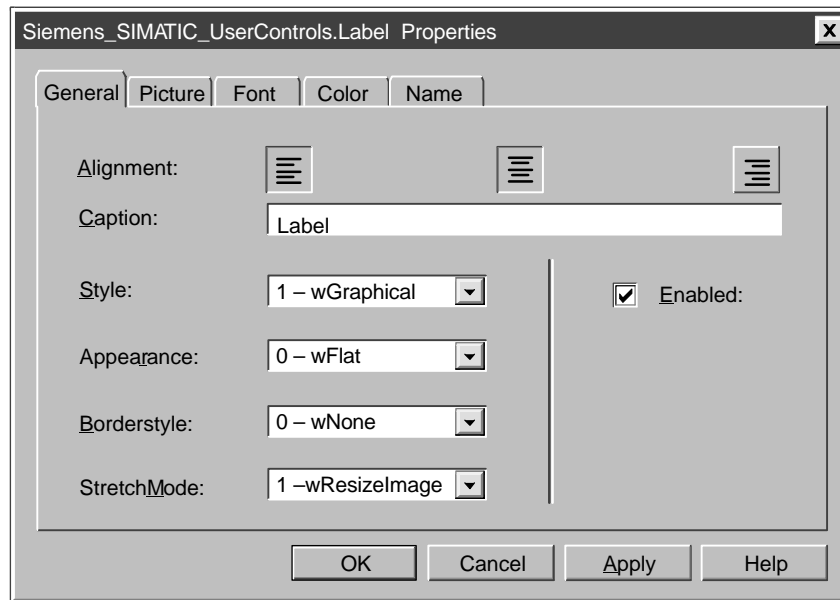


Figure 6-13 Label Control Properties (General Tab)

Defining the Picture of the Label Control (Using the Picture Tab)

The Picture tab of the Properties dialog (see Figure 6-14) allows you to browse to a picture for the Label control. You select "Picture" and then click on Browse to select the picture to be displayed for that state. Note that the Picture property can be used only if you have selected Style: Graphical on the General tab. Prefined bitmaps are provided in WinAC\WinCP\bitmaps.

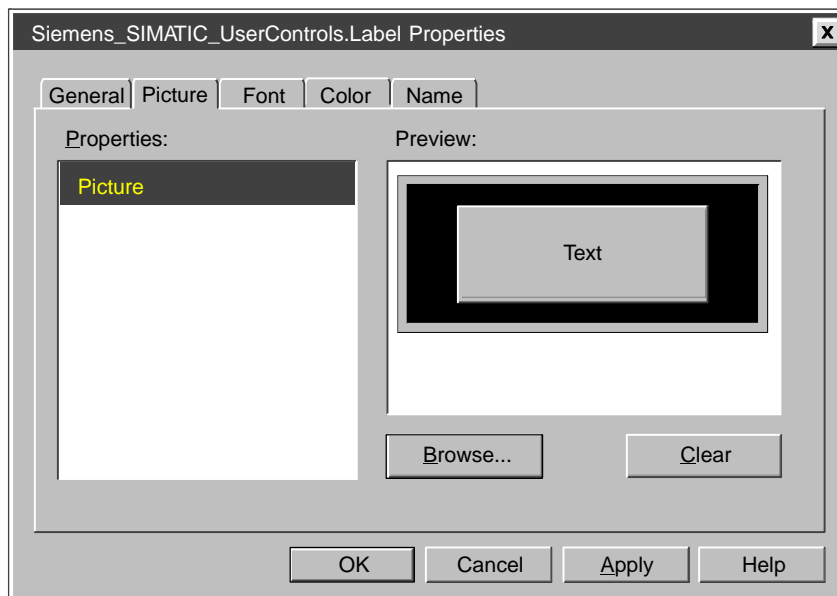


Figure 6-14 Label Control Properties (Picture Tab)

Defining the Typeface of the Label Control (Using the Font Tab)

The Font tab of the Properties dialog (see Figure 6-15) allows you to define the typeface and size for the labels of the Label control:

- “Font”: select the typeface for the label from a list of standard typefaces.
- “Size”: select the point size for the label or enter a specific point size for the label.
- “Effects”: select other typographical options (boldface, italic, underline, or strike-through) for the label.

The “Sample Text” field displays the selection of the Font property.

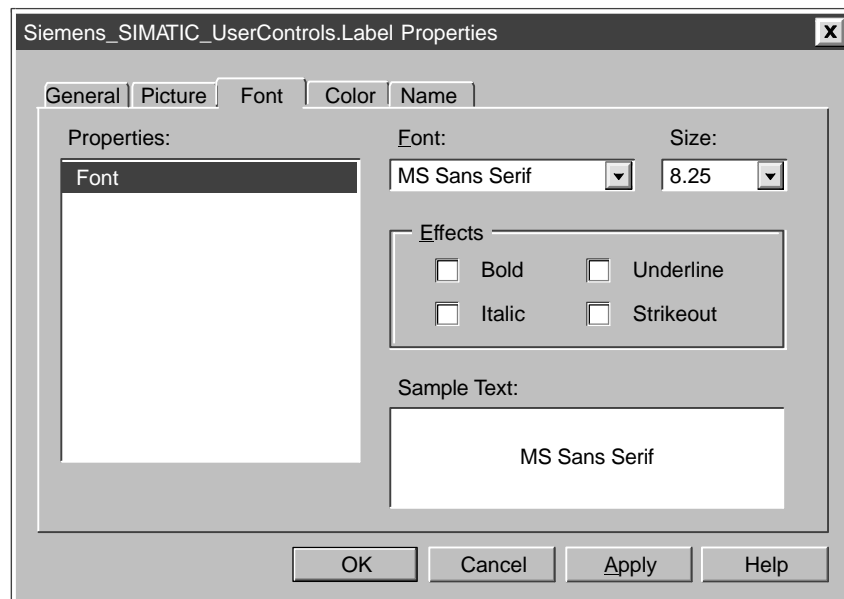


Figure 6-15 Label Control Properties (Font Tab)

Defining the Color of the Label Control (Using the Color Tab)

The Color tab of the Properties dialog (see Figure 6-16) allows you to define the colors for the background (BackColor) and for the text (ForeColor) of the Label control. You select the property ("BackColor" or "ForeColor") and then select the color to be displayed for that property from the color palette. You can choose from a palette of standard colors, or you can create custom colors.

Note

ForeColor (Text color) can be changed for both Style: Standard and Style: Graphical, but the background color (BackColor) may be hidden by the bitmap picture of the label in certain stretchmodes.

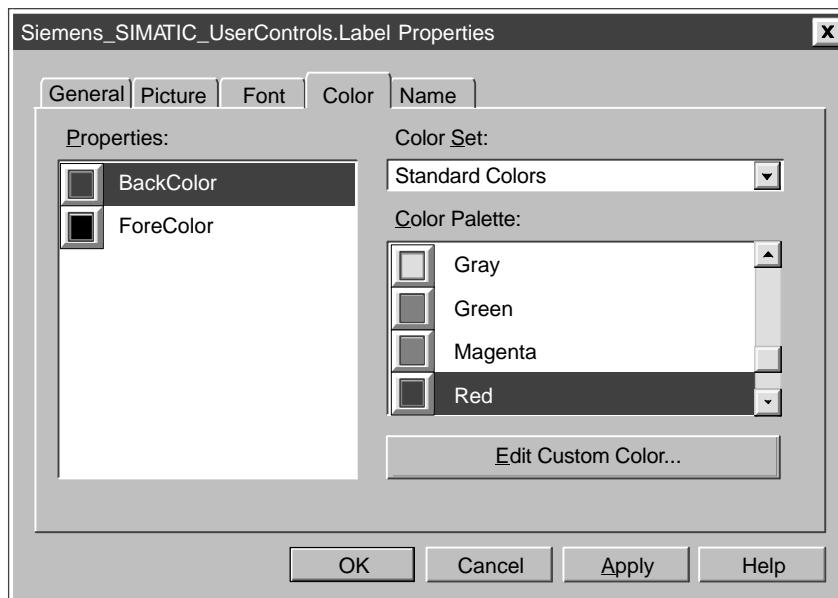


Figure 6-16 Label Control Properties (Color Tab)

Using the Name Tab

The Name tab of the Properties dialog (see Figure 6-17) allows you to assign more meaningful names to the control you have inserted into the container. This tab appears only when you are using controls in the Computing Container.

You type the new name in the “Control Name” field, and click on “Apply” or “OK”. The new name appears in the “Select Control” list in the toolbar. To open the control Properties dialog box, double-click on the desired control.

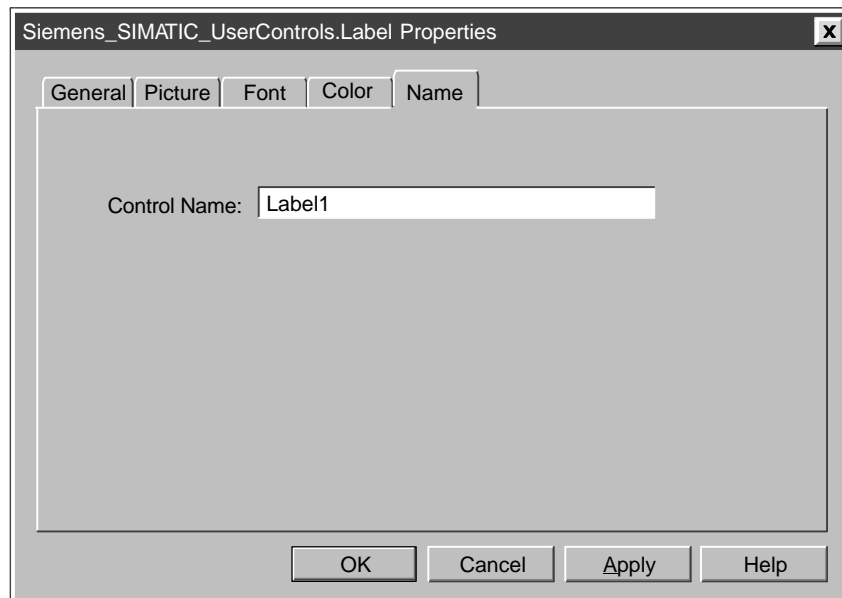


Figure 6-17 Label Control Properties (Name Tab)

6.10 Properties and Methods of the Label Control

You use the properties and methods listed in Table 6-7 to manipulate the Label control.

Table 6-7 Properties and Methods of the Label Control

Property or Method	Description	Page
AboutBox method	Displays the “About” message box for the control	B-1
Alignment property	Determines the alignment of the text	B-2
Appearance property	Determines if the control is displayed with 3D effects	B-3
BackColor property	Determines the background color for the control	B-5
BorderStyle property	Selects the border style (fixed single, or none)	B-6
Caption property	Determines the text that is displayed by the control	B-7
Enabled property	Determines whether the control responds to user-generated events	B-19
Font property	Returns a Font object for the main font of the control	B-22
ForeColor property	Determines the text color of the control	B-23
Picture property	Determines the graphic used for the control	B-28
StretchMode property	Determines the “stretch mode” of the graphic element for the control	B-38
Style property	Determines the style (standard or graphical) for the control	B-39

6.11 Events of the Label Control

The Label control responds to the events listed in Table 6-8.

Table 6-8 Events of the Edit Control

Event	Description	Page
Change event	Occurs when the value of the Value property changes	C-1
Click event	Occurs when a mouse button is pressed and released while the mouse cursor is over the control	C-1
DbClick event	Occurs when a mouse button is double-clicked while the cursor is over the control	C-2
Error event	Occurs when a property is set to an illegal value	C-2
MouseDown event	Occurs when a mouse button is pressed while the mouse cursor is over the control	C-6

Table 6-8 Events of the Edit Control, continued

Event	Description	Page
MouseMove event	Occurs when the mouse cursor moves over the control	C-7
MouseUp event	Occurs when a mouse button is released while the mouse cursor is over the control	C-8

6.12 Using the Property Pages of the Slider Control

The Slider control allows you to display process data in a visual format (as a sliding indicator) and to modify that data. You associate the slider with your process by assigning a variable (the process value) to it. You can then adjust the slider indicator in order to modify the process value; the slider also changes its indicator position automatically as the variable associated with it changes within the process.

The Slider control provides access to the memory locations of the control engine. Entering a new value in the control changes the data in the control engine.

Note

Computing does not allow you to write to timers.

Defining How the Data is to be Displayed (Using the General Tab)

The General tab of the Properties dialog box (see Figure 6-18) allows you to define the presentation of the data accessed by the Slider control.

The fields on the General tab allow you to define the following properties:

- “Style” determines the style (standard or graphical) for the control
- “Direction” sets the orientation (horizontal or vertical) of the control. See Figure 6-19.
- “StretchMode” determines the stretch mode of the graphical element for the control
- “Ticks” defines the number of interim units between the minimum and maximum values
- “SmallChange” and “LargeChange” determine the amount that the value displayed by the Slider control increases or decreases when you press an arrow key (SmallChange) or press the Page Up and Page Down keys (LargeChange).
- “KnobHeight” and “KnobWidth” determines the height and width of the indicator displayed by the control.

Using the check boxes on the General tab, you can define other operations for the control:

- “Show Min. and Max Value” check box determines whether the minimum and maximum values are displayed.
- “Enabled” checkbox determines whether the control responds to events. It does not generate events while disabled. The default setting for this option is enabled (selected).
- “Locked” checkbox determines whether the control is in a read only state. In locked state, you cannot change any values.

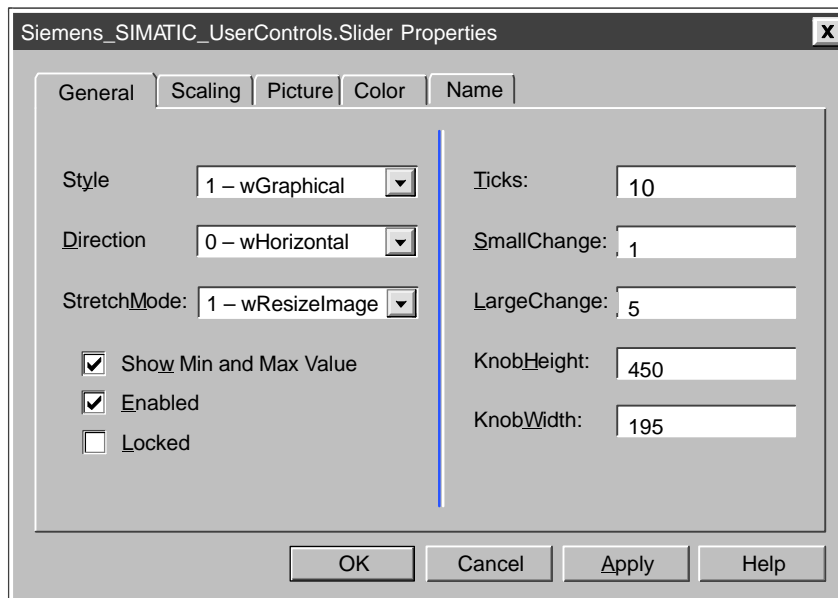


Figure 6-18 Slider Control Properties (General Tab)

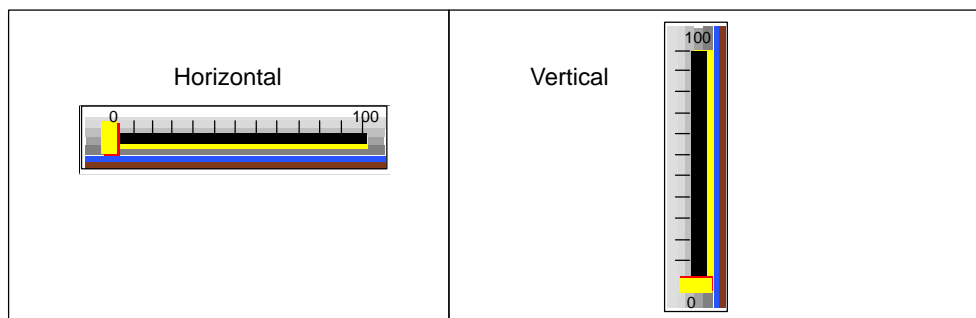


Figure 6-19 Orientation of the Slider Control

Using the Scaling Tab

The Scaling tab of the Properties dialog box (see Figure 6-21) allows you to define a scale for displaying the value in the memory location. This scaling factor is used both in reading a value from and writing a value to the control engine. You can select one of three scaling options:

- No scaling of the data (default) (0–wNoScaling)
- Scaling by formula (1–wByFormula)
- Scaling by ranges (2–wByRange)

No scaling of the data: If you choose the default, the Display Value shows an Max Value of 100 and a Min Value of 0.

Scaling by formula: If you choose to scale by formula, you enter the following information:

- Factor represents a percentage of change (scaling factor) from the value in the control engine to the value in the Slider control.
- Offset represents a fixed value to be added to the scaled result before being displayed.

The Slider control uses the following formula to calculate the scaled value:

$$(\text{Value} \times \text{Factor}) + \text{Offset} = \text{Display Value}$$

where:

Value = the value stored in the control engine

Factor = the scaling factor

Offset = the offset factor

Display Value = the value displayed in the control

When the Slider control writes data to the control engine, the inverse of the formula is used to scale the value.

Scaling by range: If you choose to scale by range transformation, you specify the upper (RawMax) and lower (RawMin) values for a source range ("Value" fields) and for a destination range ("Display Value" field). The Slider control then transforms the value from one range into the equivalent value for the other range.

These ranges define only the relationship between the data in the control engine and the data in the Slider control: if the value is above or below the ranges entered for the transformation, the transformation uses the formula to extrapolate the scaled value. The upper and lower limits are not "minimum and maximum values" for the data: there is no "limit checking" with the scaling factors.

Figure 6-20 shows the display values of the Slider control.

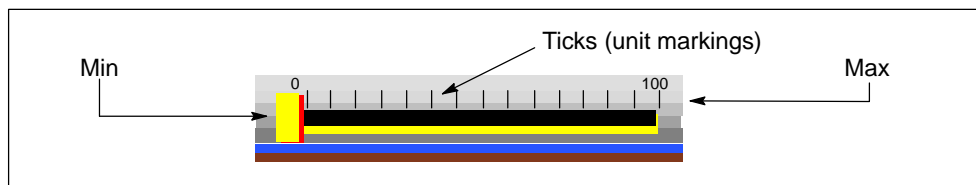


Figure 6-20 Elements of the Slider Control

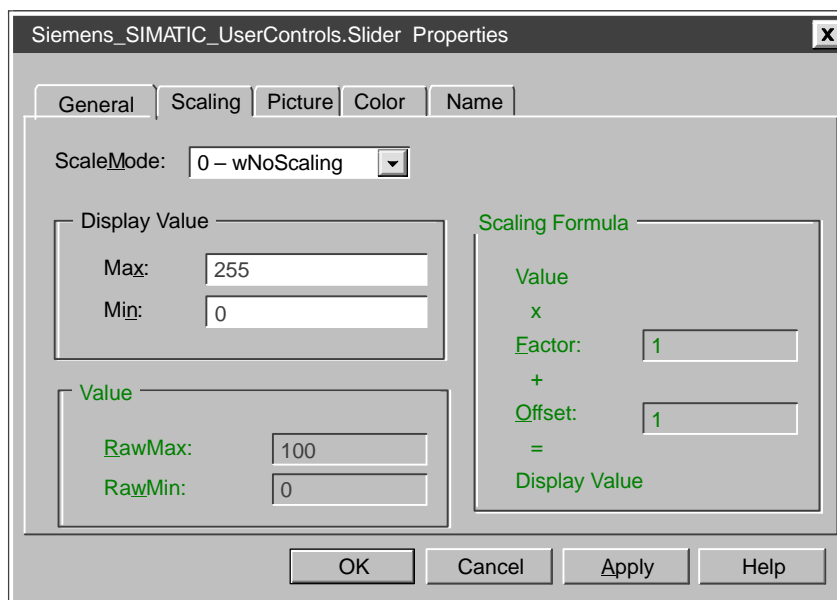


Figure 6-21 Slider Control Properties (Scaling Tab)

Defining the Picture of the Slider Control (Using the Picture Tab)

The Picture tab of the Properties dialog (see Figure 6-14) allows you to select the pictures for the Slider control. You select “KnobPicture” and then click on Browse to select the picture (graphic) used for the indicator on the control. Next, select “Picture” and then click on Browse to select the picture (graphic) used for the control. Prefined bitmaps are provided in WinAC\WinCP\bitmaps.

Note

Picture can be changed only if you have selected Style: Graphical on the General tab, but KnobPicture can be changed for both Style: Standard and Style: Graphical.

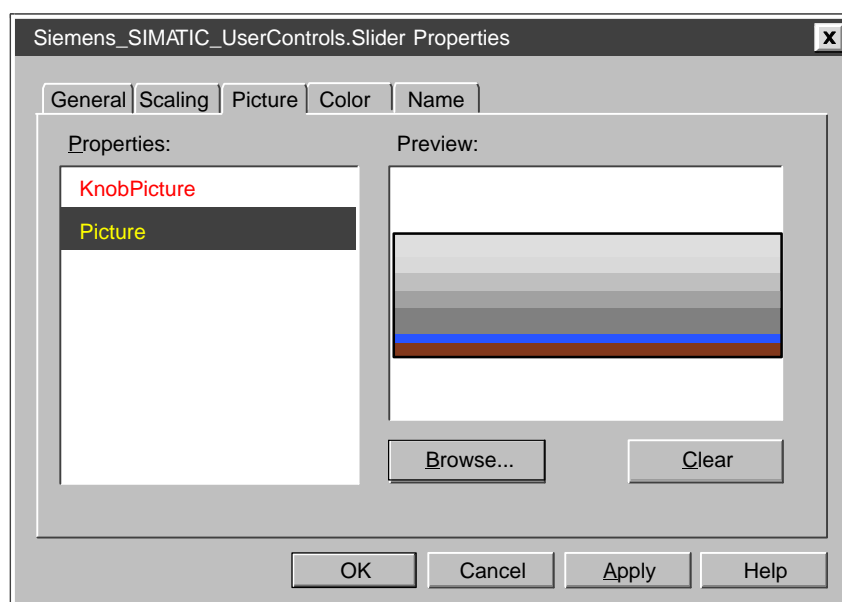


Figure 6-22 Slider Control Properties (Picture Tab)

Defining the Color of the Slider Control (Using the Color Tab)

The Color tab of the Properties dialog (see Figure 6-11) allows you to define the two colors (BackColor and ForeColor) and for the text of the Slider control. You select the Property ("Back Color" or "Fore Color") and then select the color to be displayed for that property from the color palette. You can choose from a palette of standard colors, or you can create custom colors.

- "BackColor" defines the background color of the control.
- "ForeColor" defines the color used to display text and graphics in an object.

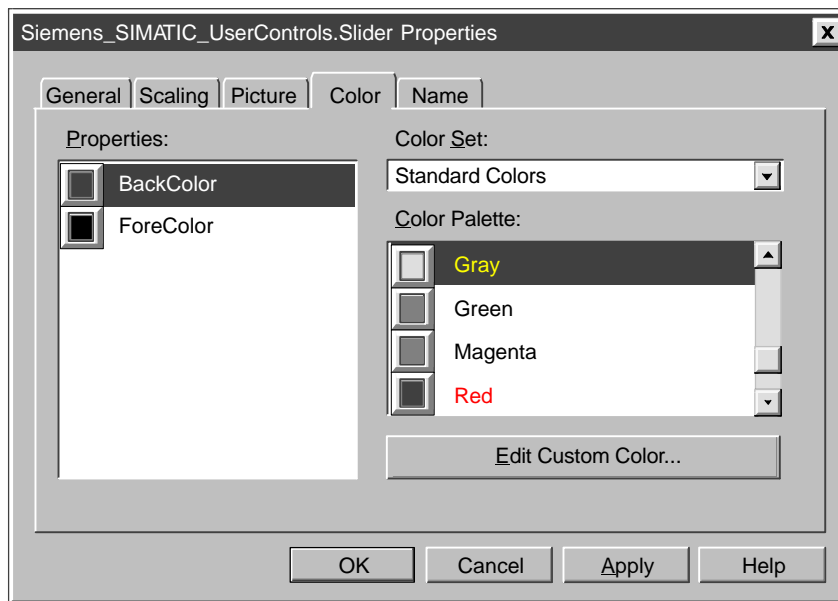


Figure 6-23 Slider Control Properties (Color Tab)

Note

BackColor can be changed only if you have selected Style: Standard on the General tab, but ForeColor can be changed for both Style: Standard and Style: Graphical.

Using the Name Tab

The Name tab of the Properties dialog (see Figure 6-24) allows you to assign more meaningful names to the control you have inserted into the container. This tab appears only when you are using controls in the Computing Container.

You type the new name in the “Control Name” field, and click on “Apply” or “OK”. The new name appears in the “Select Control” list in the toolbar. To open the control Properties dialog box, double-click on the desired control.

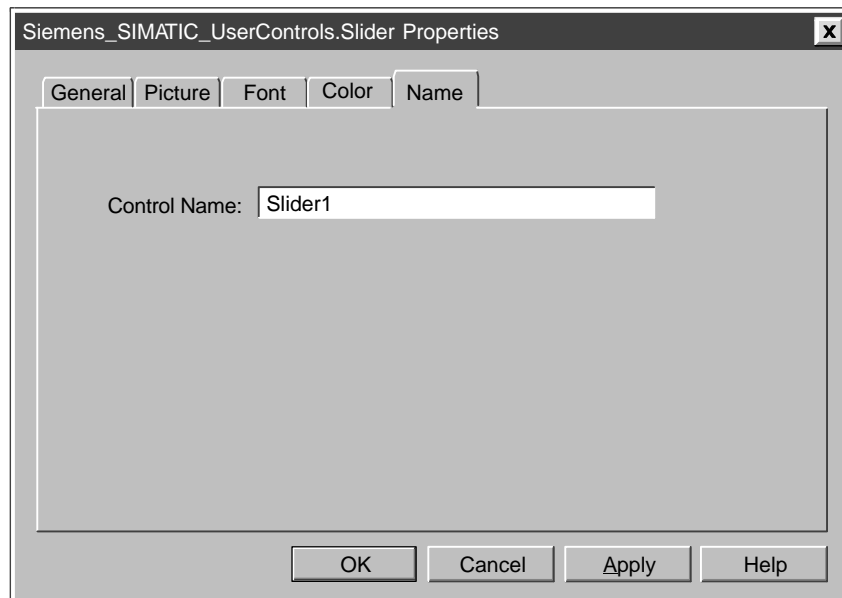


Figure 6-24 Slider Control Properties (Name Tab)

6.13 Properties and Methods of the Slider Control

You use the properties and methods listed in Table 6-9 to manipulate the Slider control.

Table 6-9 Properties and Methods of the Slider Control

Property or Method	Description	Page
AboutBox method	Displays the “About” message box for the control	B-1
BackColor property	Determines the background color for the control	B-5
Direction property	Sets the orientation (horizontal or vertical)	B-14
Display Value property	Returns the scaled value for the control	B-18
Enabled property	Determines whether the control responds to user-generated events	B-19
Factor property	Specifies the scaling factor used when the scale-by-formula option has been enabled (used with ScaleMode property)	B-20
ForeColor property	Determines the foreground color for the control	B-23
KnobHeight property	Determines the height of the indicator displayed by the control	B-24
KnobPicture property	Determines the graphical element (picture) to be used as the indicator for the control	B-24
KnobWidth property	Determines the width of the indicator displayed by the control	B-25
LargeChange property	Determines how far the slider indicator moves when the control has focus and you press the Page Up or Page Down key	B-25
Locked property	Sets the control to a read-only state. By default, the control is not in locked mode, so the user can enter numbers.	B-26
Max property	Returns/sets the maximum scaled value of the control if the ScaleMode property is wByRange or wScaleNone	B-26
Min property	Returns/sets the minimum scaled value of the control if the ScaleMode property is wByRange or wScaleNone	B-26
Offset property	Specifies the offset used when the scale-by-formula option has been enabled (Used with ScaleMode property)	B-27
Picture property	Determines the graphical element (picture) to be used for the control	B-28
RawMax property	Determines the maximum raw value of the control (if the ScaleMode property is set to “wByRange”)	B-32
RawMin property	Defines the lower value of the source range for scaling a value. The ScaleMode property must be set to “wByRange”.	B-32

Table 6-9 Properties and Methods of the Slider Control, Fortsetzung

Property or Method	Description	Page
ScaleMode property	Specifies the scaling mode to be used for scaling values	B-34
ShowMinMax property	Specifies whether the control displays the range (minimum and maximum) of values	B-36
Style property	Determines the style (standard or graphical) of the control	B-39
SmallChange property	Determines how far the slider indicator moves when the control has focus and you press the up/down or right/left arrow keys	B-37
StretchMode property	Determines the “stretch mode” of the graphic element for the control	B-38
Ticks property	Sets the number of ticks (unit markers)	B-40
Value property	Contains the value that is linked to the control engine	B-43

6.14 Events of the Slider Control

The control responds to the events listed in Table 6-10.

Table 6-10 Events of the Slider Control

Event	Description	Page
Change event	Occurs when the value of the Value property changes	C-1
Click event	Occurs when a mouse button is pressed and released while the mouse cursor is over the control	C-1
DbClick event	Occurs when a mouse button is double-clicked while the cursor is over the control	C-2
Error event	Occurs when a property is set to an illegal value	C-2
KeyDown event	Occurs when the user presses a key while the control has the focus	C-3
KeyPress event	Occurs when an ANSI key is pressed and released while the control has the focus	C-4
KeyUp event	Occurs when a key is released while the control has the focus	C-5
MouseDown event	Occurs when a mouse button is pressed while the mouse cursor is over the control	C-6
MouseMove event	Occurs when the mouse cursor moves over the control	C-7
MouseUp event	Occurs when a mouse button is released while the mouse cursor is over the control	C-8

S7 Diagnostic Buffer Control (DBuffer)

7

The diagnostics buffer of the S7 controllers is a ring buffer that contains entries written by the operating system of the S7 controller. Each entry contains information about a specific diagnostic event. These events are displayed in the order in which they are generated, with the most recent event listed first. The DBuffer control allows your program to access the diagnostics buffer and display the events.

For more information about the diagnostics buffer, refer to the online help for the STEP 7 programming software and to the documentation for the S7 controllers.

Section	Description	Page
7.1	Using the S7 Diagnostics Buffer	7-2
7.2	Configuring the DBuffer Control	7-4
7.3	Properties, Methods, and Events of the DBuffer Control	7-7

7.1 Accessing the S7 Diagnostics Buffer

As shown in Figure 7-1, you can use the DBuffer control to access the diagnostics buffer of an S7 controller without going through either STEP 7 or the Data control. The DBuffer displays the following information from the diagnostics buffer:

- The control lists the diagnostic events which were generated by the controller, with the date and time that the event occurred. The user can select any event in the list to display additional information.
- The control provides detailed information about each event. The user can select to display either:
 - Textual description which provides information such as the address of the instruction that caused the event and the mode transition that was caused by the event
 - Hexadecimal values for the 20 bytes of the diagnostics event
- The control displays the hexadecimal event ID.

If you enable the control to sort the events, you can use the buttons in the upper panel to sort the events by the type of event (either the description or by the hexadecimal event ID) or by the sequence that the event occurred (which is determined by the time and date when the event was generated).

The DBuffer control does not use the Data control to access the control engine. You configure the properties of the DBuffer control for the control engine and for other properties.

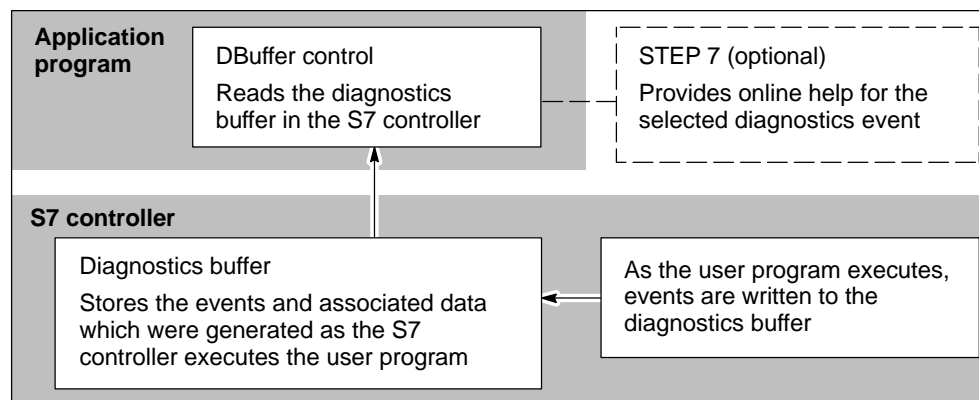


Figure 7-1 Accessing the diagnostics Buffer of an S7 Controller

Note

The DBuffer does not automatically read and update the diagnostic buffer in the controller: the user must manually request an update by clicking on the “Update” button.

As shown in Figure 7-2, the DBuffer control provides the following elements:

- Upper panel: lists the events of the diagnostics buffer of the controller, including the number (starting with the most recent), the date and time of the diagnostic event, and a short description.
- Lower panel: provides a detailed description of a selected event listed in the upper panel. This information includes the name and event number of the event, additional description (such as the address of the instruction that caused the event), and the current event state.
- Update button: reads the diagnostics buffer and updates the information displayed by the control. The DBuffer control does not automatically read the S7 diagnostics buffer and update the events.
- Language: selects the language (German, English, French, Italian, or Spanish) for the descriptions of the events. The language of the column headings and the buttons (German, English, or French) is selected by the Panel control or the container.
- Help on Event button: displays the STEP 7 online help for the selected diagnostic event.

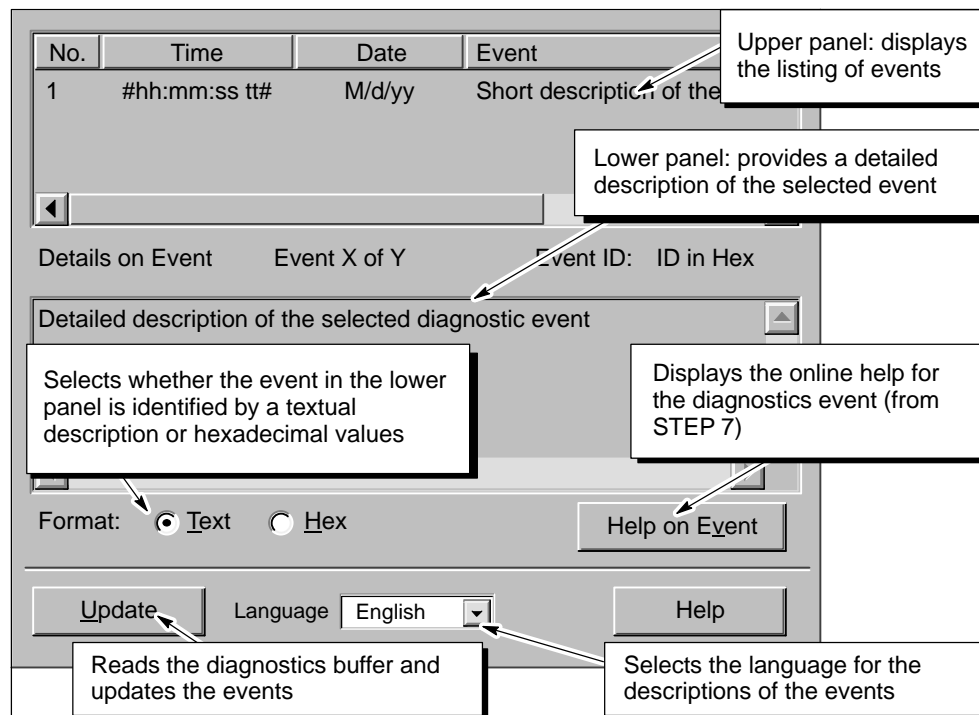


Figure 7-2 Elements of the DBuffer Control

7.2 Configuring the DBuffer Control

Figure 7-3 shows the dialog box for configuring the “Properties” dialog box of the DBuffer Control. To access the “Properties” dialog box, click the right button of the mouse and select the **DB1 Properties** command. You can also access the properties of the control with your application program. With “Diagnostic Buffer” tab, you can set the following properties:

- **Enable Sort:** When selected, this property (EnableSort) allows the user to sort the diagnostic events by clicking on the headings (No., Time, Date, or Event) for the diagnostic buffer. See Figure 7-2.
- **Text or Hex(adecimal):** This property (FormatDisplay) displays the information about the diagnostic event in either hexadecimal numbers or as text.
- **Control Engine:** This property (ControlEngine) selects the control engine. For WinLC, enter `winLC`, or `wcs7=3` for a slot PLC such as the CPU 416-2 DP ISA.

You can also test the connection to the control engine by clicking on the “Connection Test” button.

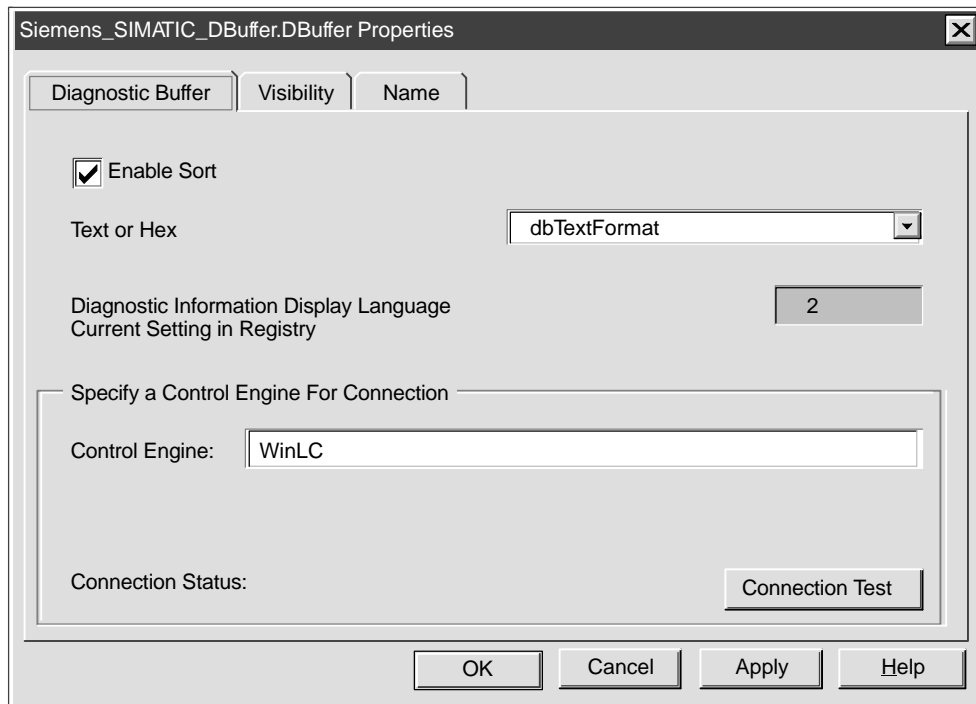


Figure 7-3 “Diagnostics Buffer” Tab of the DBuffer Control

Figure 7-4 shows “Visibility” tab of the “Properties” dialog box for the DBuffer control. With this tab, you can select the various elements of the DBuffer control to be displayed:

- Display Upper Panel: When selected, this property (DisplayUpperPanel) displays the upper panel of the diagnostic buffer. This panel displays the listing of the diagnostic buffer.
- Display Lower Panel: When selected, this property (DisplayLowerPanel) displays the lower panel of the diagnostic buffer. This panel provides the detailed description about the selected event.
- Display Format Buttons: When selected, this property (DisplayFormatButtons) displays format buttons for selecting whether the data for the event is displayed as text or as a hexadecimal value.
- Display Help-On-Event Button: When selected, this property (DisplayHelpOnEvent) displays the “Help on Event” button that allows the user to get online help for the selected diagnostic event. (STEP 7 must be installed on the same computer as the DBuffer control.)
- Display Help Button: When selected, this property (DisplayHelpButton) displays the “Help” button that allows the user to get online help about the DBuffer control.
- Display Update Button: When selected, this property (DisplayUpdateButton) displays the “Update” button that reads the diagnostic buffer in the S7 controller.

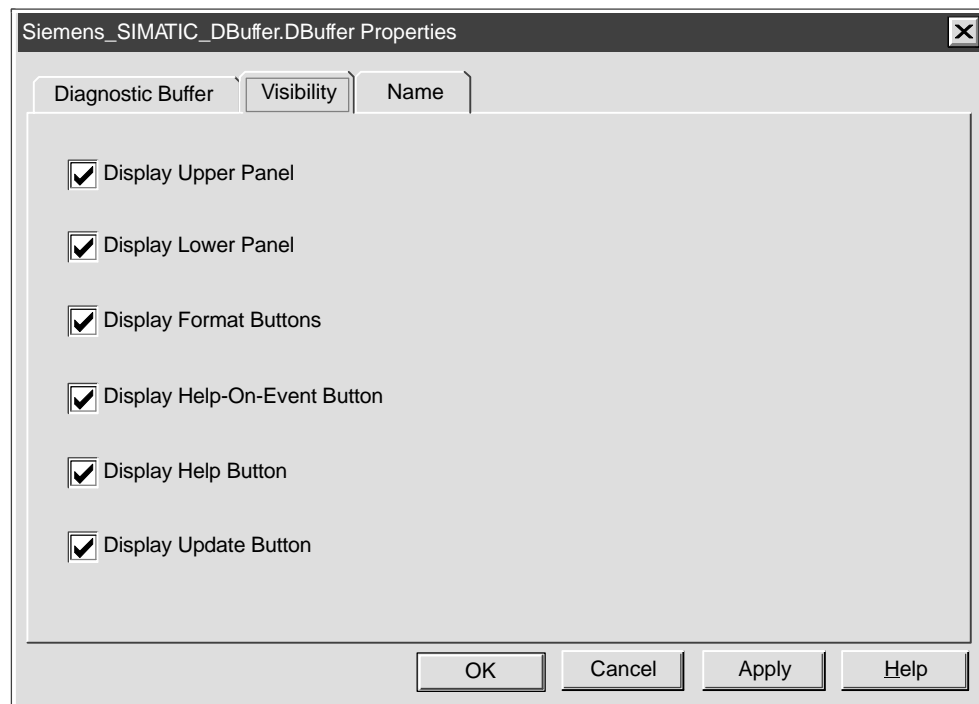


Figure 7-4 “Visibility” Tab of the DBuffer Control

Using the Name Tab

The Name tab of the Properties dialog (see Figure 7-5) allows you to assign more meaningful names to the control you have inserted into the container. This tab appears only when you are using controls in the Computing Container.

You type the new name in the “Control Name” field, and click on “Apply” or “OK”. The new name appears in the “Select Control” list in the toolbar. To open the control Properties dialog box, double-click on the desired control.

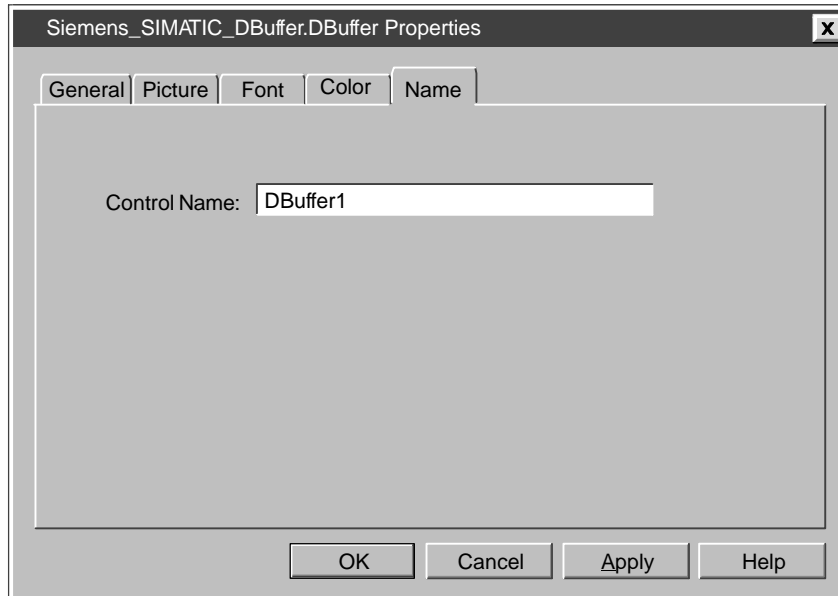


Figure 7-5 DBuffer Control Properties (Name Tab)

7.3 Properties and Methods of the DBuffer Control

You use the properties and methods listed in Table 7-1 to manipulate the DBuffer control.

Table 7-1 Properties and Methods of the DBuffer Control

Property or Method	Description	Page
bDiagBuffOK property	Verifies the connection to the diagnostic buffer	B-5
bEngineConnected property	Verifies the connection to the control engine	B-6
ControlEngine property	Stores the filename for the control engine	B-10
DisplayFormatButtons property	Shows or hides the "Text" and "Hexadecimal" buttons	B-15
DisplayHelpButton property	Shows or hides the "Help" button	B-15
DisplayHelpOnEventButton property	Shows or hides the "Help on Event" button	B-16
DisplayLowerPanel property	Shows or hides the lower panel of the DBuffer control	B-17
DisplayUpdateButton property	Shows or hides the "Update" button	B-17
DisplayUpperPanel property	Shows or hides the upper panel of the DBuffer control	B-18
EnableSort property	Enables or disables the sorting for the columns in the upper panel of the DBuffer control	B-19
FormatDisplay property	Changes the formatting for the additional information for a particular event	B-23
PopUpHelp method	Displays the online help for the DBuffer control	B-29
PopUpHelpOnEvent method	Displays the online help (from STEP 7) for the selected diagnostic event	B-29
SelectEvent method	Selects a specific event from the upper panel of the DBuffer control	B-35
Update method	Reads the diagnostic buffer in the control engine and updates the events listed in the control	B-42

Designing Simple Process Forms with the SoftContainer

8

Chapter Overview

Computing provides an OLE container application (SoftContainer) for receiving and displaying the data from the control engine. Using this container, you can insert your own third-party controls or the SIMATIC controls into a process form.

This chapter provides information about inserting and positioning the controls in the container. For more information about the specific SIMATIC controls, refer to the following chapters:

- For information about the Data control, see Chapter 5.
- For information about the diagnostic buffer control (DBuffer), see Chapter 7.
- For information about the other SIMATIC controls (Button control, Edit control, Label control and Slider control), see Chapters 6, 7, 8 and 9.

Section	Description	Page
8.1	Starting the Computing SoftContainer	8-2
8.2	Creating a Process Form	8-4
8.3	Switching from Design Mode to Run Mode	8-6
8.4	Saving Your Process Form	8-8

8.1 Starting the Computing SoftContainer

Computing includes a container for the various SIMATIC controls. To create a container, select the **Simatic ► PC Based Control ► Computing Softcontainer** menu command from the Start menu. You can also double-click on the icon for Computing. Figure 8-1 shows a sample container, which includes the following elements:

- The toolbar contains buttons for accessing common functions (such as for opening a process form, or for cutting and pasting). It also contains the icons for the SIMATIC controls provided by Computing.
- The toolbar also contains a field that contains the name of the selected control. You can use the drop-down list box to select the controls in the process form.
- The status bar contains information about the operating mode of S7Soft container (Design or Run). The status bar also provides information about which control has been selected, including its size and location within the process form.
- A default process form (S7Soft1) to contain the controls.

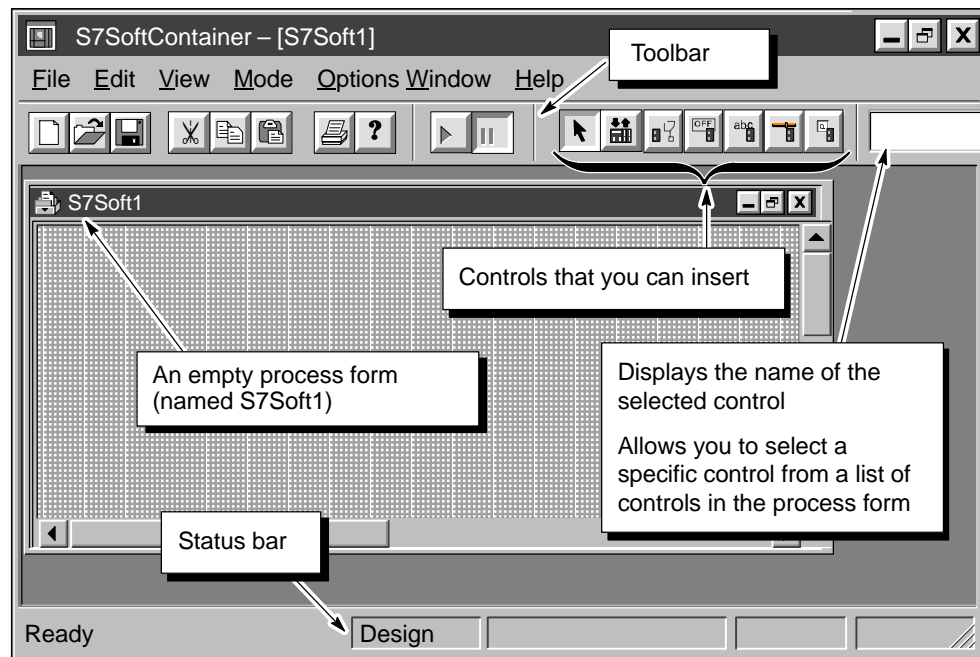


Figure 8-1 Container with the Default Process Form

Using the Snap Grid and the Status Bar

The container provides a snap grid to help position or size the controls. As shown in Figure 8-2, the status bar provides information about the selected control:

- Position information. The status bar shows the current coordinates of the snap grid for the control. (This information is displayed even when the snap grid is disabled.)
- Size information. The status bar shows the size of the control (width x height).

By selecting several controls, you can use the information in the status bar to adjust the size or position of the controls.

You can turn the status bar and the snap grid on or off:

- To enable the snap grid, select the **View ► Snap Grid** menu command. The snap grid is enabled when the menu displays a check mark.
- To display the status bar, select the **View ► Status Bar** menu command. The status bar is displayed when the menu displays a check mark.

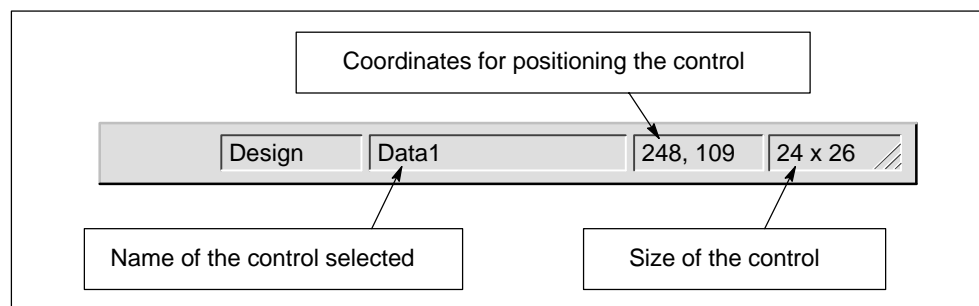


Figure 8-2 Elements of the Status Bar

8.2 Creating a Process Form

A “process form” is a document or file created with the SoftContainer. It contains the ActiveX controls used for monitoring and modifying data in the control engine.

Inserting a SIMATIC Control in the Process Form

Refer to Figure 8-3 and use the following procedure to insert a SIMATIC control into your process form:

1. In the toolbar, select the control to be inserted by clicking on the icon for the control. (Figure 8-3 shows a Data control being inserted into the process form.)
2. Use the mouse to move the arrow pointer to the open process form. Inside the process form, the arrow pointer changes to a cross-hair pointer.
3. Click the left mouse button to insert the control that you selected.

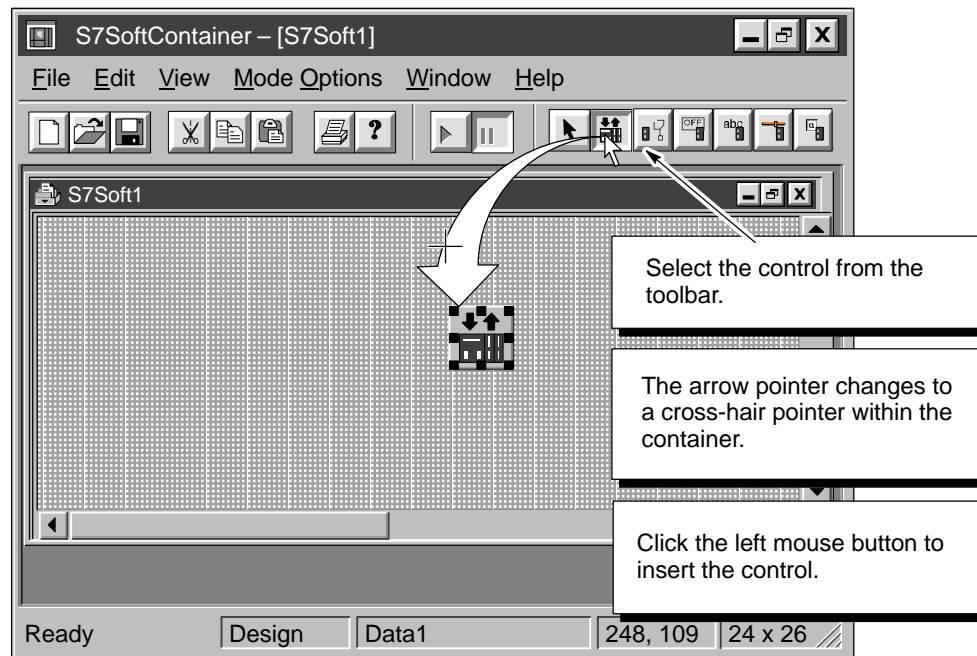


Figure 8-3 Inserting a Control from the Toolbar

Inserting Third-Party Controls into the Process Form

You can use other ActiveX controls than the SIMATIC controls in your process form. Use the following procedure to insert a custom or a third-party control into your process form:

1. Select the **Edit ► Insert Control** menu command to display the “Insert Control” dialog box.
2. As shown in Figure 8-4, select the custom or third party control to be added to the process form. (You can add the control to the toolbar of the SoftContainer by selecting the “Add control to toolbar” check box. This allows you to use the icon in the toolbar for inserting the control into the process form.)
3. Click on the “OK” button and insert the control into the process form.

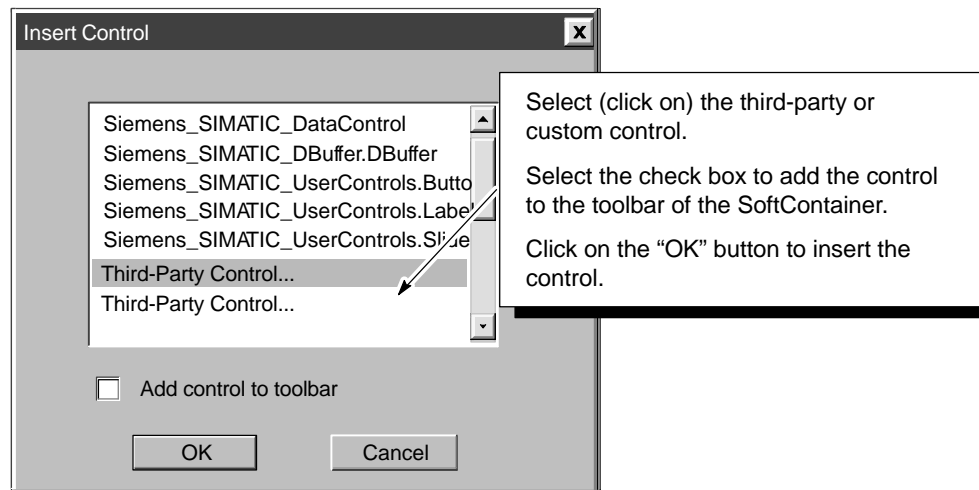


Figure 8-4 Inserting a Third-Party Control into the Process Form

Configuring the Data Control

Before you can connect to the control engine, you must configure the Data control for communicating with a control engine. To configure the Data control, see Chapter 5.

Note

The Diagnostic Buffer control does not use Data control to connect to the PLC.

8.3 Switching from Design Mode to Run Mode

When you switch the SoftContainer from Design mode to Run mode, you connect the controls to the control engine. These operating modes define the operation of the SoftContainer only and do not apply to the operating modes for the control engine.



Warning

After you connect a SIMATIC or a third-party control to your process data by assigning a variable to its Value property, any changes that you make to the value displayed in the control are immediately applied to your process data.

Altering process data can cause unpredictable equipment operation, which could result in death or serious injury to personnel, and/or damage to equipment.

Do not perform the exercises in this chapter when your control engine is connected to a real process. These exercises are for practice only. Do not modify any data that could cause process equipment to operate erratically. Always install a physical emergency stop circuit for your machine or process.

Changing the Operating Mode of the SoftContainer

The SoftContainer provides two operating modes that are not related to the operating modes of the control engine:

- Design mode allows you to insert and modify the controls in the process form. You can change the properties in Design mode.
- Run mode connects the controls to the control engine. You can modify values in the control engine, but you cannot move or modify the properties for the controls.

Note

The controls must have an active control engine (such as WinLC) in order to access the process data. Be sure that the control engine is running before you switch the SoftContainer from Design mode to Run mode.

Use the following procedure to change the operating mode of the container:

1. Make certain that the control engine (such as WinLC) is running. For information about starting the control engine, refer to the process formation for the control engine.

2. Click on the Run button to change from Design mode to Run mode. See Figure 8-5. The status bar shows that the container is in Run mode.

Notice that the Data control becomes invisible in Run mode.

In Run mode, you can use the controls to monitor or to modify the values stored in the control engine. To return to Design mode, click on the Design button.

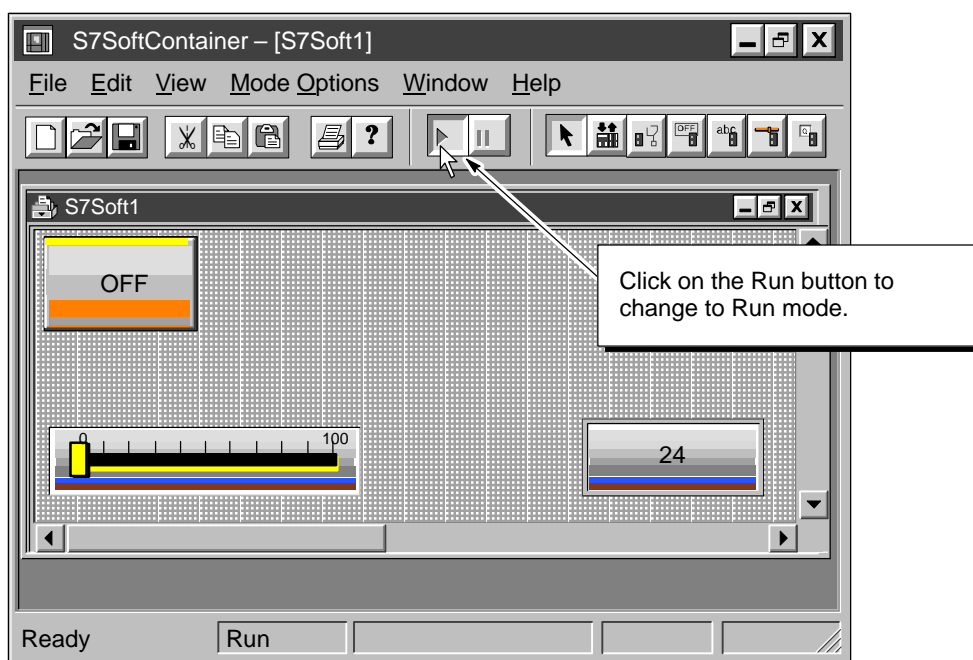


Figure 8-5 Changing the Container to Run Mode

8.4 Saving Your Process Form

You can save the process forms you create under any name in any directory that is convenient. Use the following procedure to save your process form:

1. Select the **File ► Save As** menu command to display the Save As dialog box. See Figure 8-6.
2. Enter the name for the process form.
3. Choose a directory for storing the process form. The default directory for storing process forms is in the Computing directory (WinCP).
4. Click on the “Save” button.

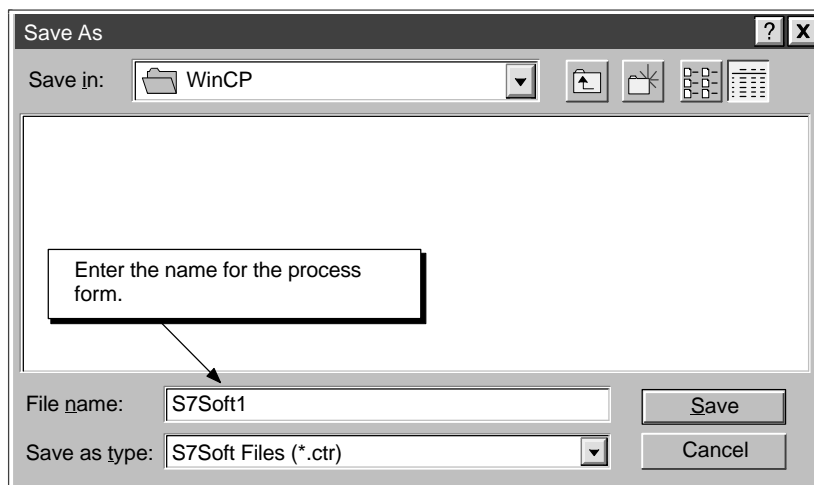


Figure 8-6 Saving a Process Form for Computing

Creating Tag Files with the TagFile Configurator

9

The TagFile Configurator creates tag files that allow you to use symbols for the memory locations being accessed in the control engine. The tag file provides a source of symbolic information for memory locations and control engines. Linking to a tag file allows you to use symbolic names instead of absolute addresses when assigning variables in the Data control.

Multiple STEP 7 programs can be mapped into a single tag file, with each program providing access to a different computer and control engine. This allows Computing to access data from different computers and control engines simultaneously.

Using a tag file also enables the completion aid in the Data control: entering a “.” (dot or period) displays a list of valid tag files or symbols. You can also click on the Browse button to navigate to the symbol.

Section	Description	Page
9.1	Connecting to Multiple Control Engines over DCOM	9-2
9.2	Using Symbols to Access Data on Multiple Control Engines	9-5
9.3	Creating a Tag File	9-6
9.4	Configuring a Tag File for Local or Remote Access to a Control Engine	9-10
9.5	Changing the Control Engine Symbol Name in the Tag File Editor	9-13

9.1 Connecting to Multiple Control Engines over DCOM

As shown in Figure 9-1, you can use DCOM to connect your program to control engines residing on several different computers. You use the TagFile Configurator to create a tag file that identifies symbol names for the variables of the different control engines.

Multiple STEP 7 programs can be mapped into a single tag file, with each program providing access to a different computer and control engine. This allows the Data control to access data from different computers and control engines simultaneously.

See Section 9.3 for information about creating a tag file. See Section 9.4 for information about configuring the control engine for local or remote access.

Note

To maintain the name of a remote computer within a STEP 7 project, create a station name in STEP 7 with the following components:

- @ (Starting the station name with “@” signals the TagFile Configurator that this station name refers to a remote computer.)
- Name of the remote computer (DCOM identifier)

For example (as shown in Figure 9-1): @PC 2

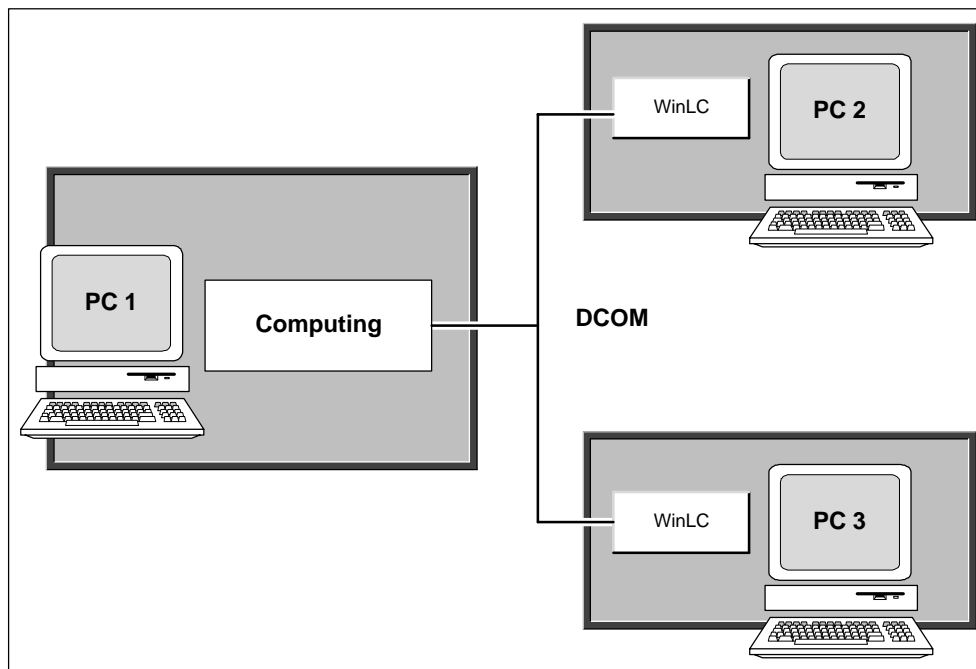


Figure 9-1 Connecting to Multiple Control Engines over DCOM

The TagFile Configurator creates a tag file that provides a source of symbolic information for the memory locations and control engines. The tag file contains the following information:

- Computer name: identifies the computer where the control engine resides, either the local computer or a networked computer. If you configured the station name in STEP 7 to begin with an “@” symbol, the TagFile Configurator recognizes the station name as a DCOM address name for the computer that is running the control engine.
- Control engine: identifies the control engine to be accessed by the Data control. This information typically comes from the CPU configured in STEP 7. For example, `winLC` (for WinLC), `wcs7=3` (for a slot PLC such as the CPU 416-2 DP ISA), or `wcs7=xx,a,b` (for other PLCs on the network, where `xx` is the node address of the PLC, `a` is the rack number, and `b` is the slot number). See Appendix G for more information about control engine strings.

Creating a Tag File with Multiple Control Engines

Use the following procedure to create a tag file that contains symbols for several control engines:

1. Use the Start menu (**Start ▶ Simatic ▶ PC Based Control ▶ TagFile Configurator**) to open the TagFile Configurator.
The TagFile Configurator is opened with a new (empty) file.
2. Select the **Insert ▶ Program** menu command to choose the programs to be added to the tag file.
3. Select several programs and add them to the tag file by clicking on the “→” button. Figure 9-2 shows two programs (“Master_Mixer” on PC 2 and “My_Drain” on PC 3) forming a single tag file. When you have added the programs for the tag file, click on the “OK” button.

The station names must begin with an “@” (such as “@PC 2”) for the TagFile Configurator to recognize that the connection for the control engine will be over DCOM.

4. Use the **File ▶ Save As...** menu command to save (and rename) the tag file.

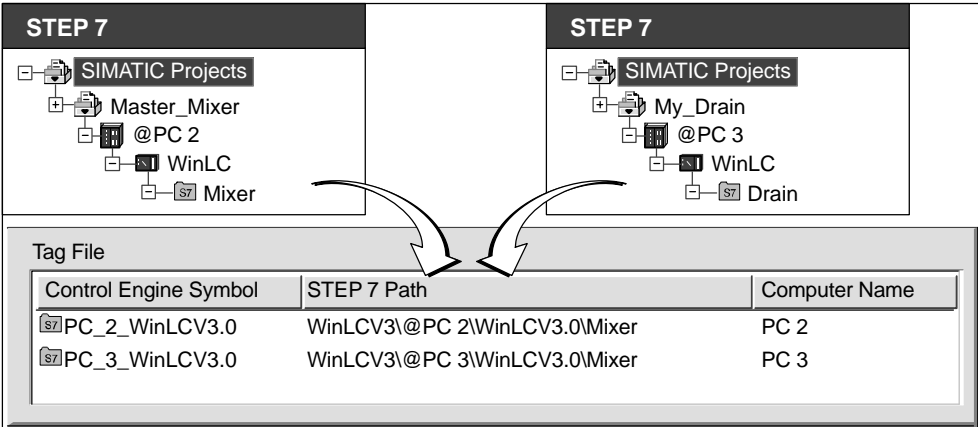


Figure 9-2 Creating a Tag File for Multiple Control Engines

Configuring the Data Control for Multiple Control Engines

Use the following procedure to configure the Data control to use a tag file that contains symbols for several control engines:

1. Use the Start menu (**Start ► Simatic ► PC Based Control ► SIMATIC Computing**) to open the Computing container.
2. Insert a Data control.
3. Double-click on the Data control (or use the **Edit ► Properties** menu command) to display the “Properties” dialog box for the Data control.
4. Click on the “Engine” tab to display the configuration choices. See Figure 9-3.
5. Select the “Connection via Tag Source” option.
6. Click on the “Browse” button and select the tag file that contains the symbols for multiple control engines.
7. Click on the “OK” button to configure the Data control for communicating with control engines running on different computers.

All of the controls that you insert and connect through this Data control will access the specified variables in the various remote control engines.

Note

You must have configured the different computers for DCOM. See Sections E.2 and E.3 for information about configuring the server and client computers for DCOM.

For additional information about DCOM, refer to the online help for Windows NT.

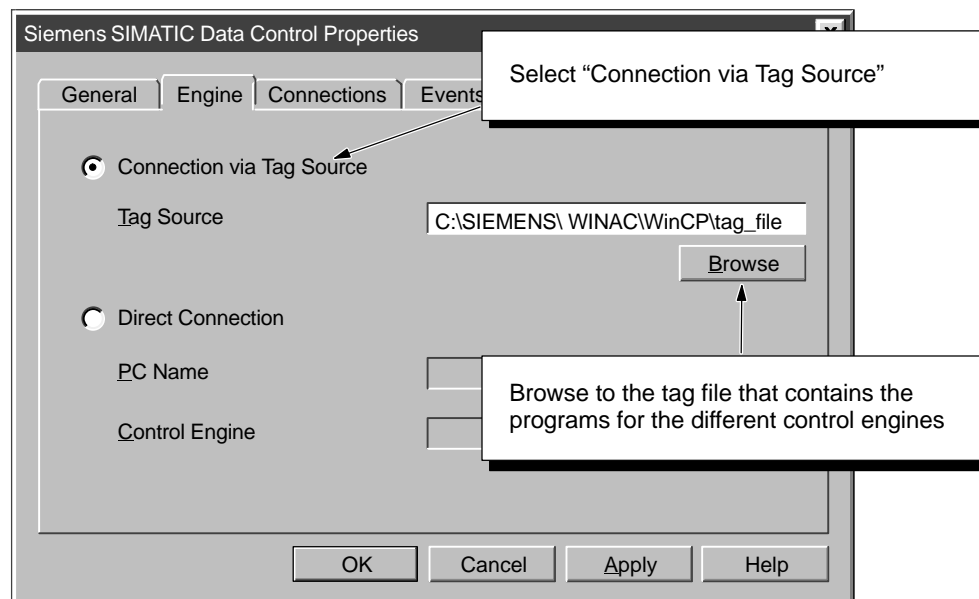


Figure 9-3 Configuring the Data Control for Multiple Control Engines

9.2 Using Symbols to Access Data in the Control Engine

A tag file provides a source of symbolic information for memory locations and control engines. Linking to a tag file allows you to use symbolic names instead of absolute addresses when assigning variables in the Data control provided with the Computing software. The TagFile Configurator creates a tag file (*.tsd) that provides a source of symbolic information for the memory locations and control engines. See Figure 9-4.

Note

The TagFile Configurator must be running on a computer that also has STEP 7 installed in order for you to insert STEP 7 programs for using symbolic addresses. (STEP 7 does not have to be installed for you to insert a control engine without symbolic information into the tag file.)

To use the tag file with a Data control, the tag file must reside on a computer that can be accessed by the Data control.

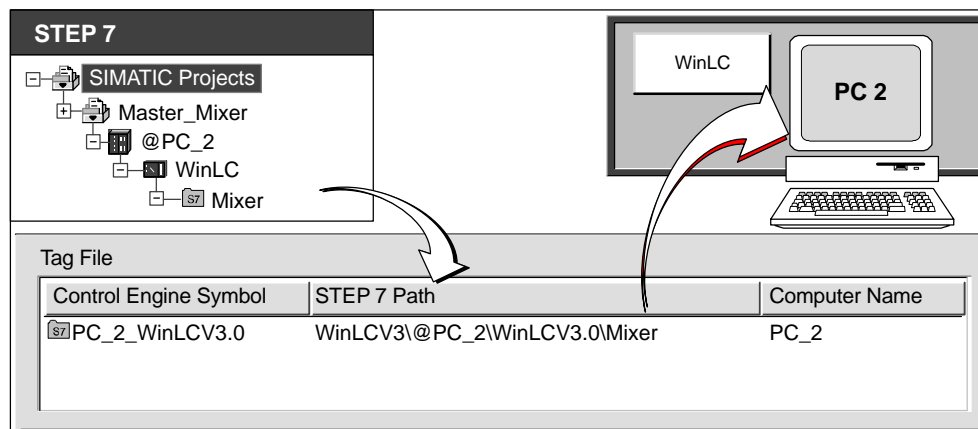


Figure 9-4 Using Symbols to Access Data in the Control Engine

The tag file includes the following elements:

- Control engine: identifies the control engine to be accessed by the Data control. This information typically comes from the PLC configured in STEP 7. See Appendix G for more information about control engine identifiers.
- Computer name: identifies the computer where the control engine resides, either the local computer or a networked computer. If you configured the station name in STEP 7 to begin with an "@" symbol, the TagFile Configurator recognizes the station name as a DCOM address name for the computer that is running the control engine.

9.3 Creating a Tag File

The TagFile Configurator must be running on a computer that also has STEP 7 installed for you to insert STEP 7 programs into a tag file that will allow you to use symbolic addresses. (STEP 7 does not have to be installed for you to insert a control engine without symbolic information into the tag file.) To use the tag file with a Data control, the tag file must reside on a computer that can be accessed by the Data control.

See Section 9.4 for information about configuring the control engine for local or remote access.

Opening the TagFile Configurator

Use the following procedure to start the TagFile Configurator:

1. Select the **Start ► Simatic ► PC Based Control ► TagFile Configurator** menu command.

The TagFile Configurator is opened with a new (empty) file.

Figure 9-5 shows the TagFile Configurator with an empty tag file open. You insert STEP 7 programs and control engines into the tag file.

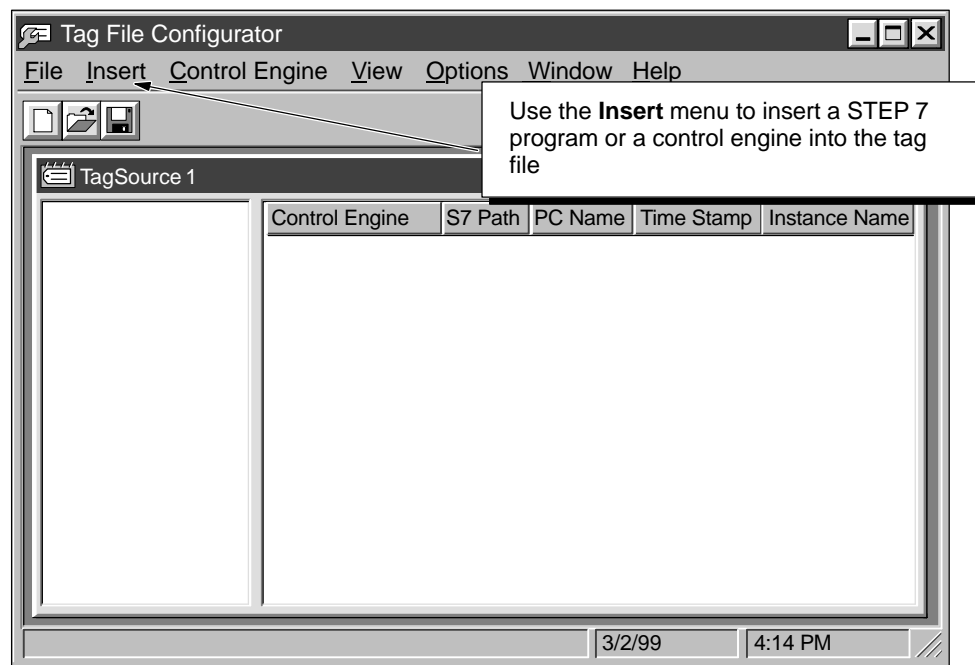


Figure 9-5 TagFile Configurator

Inserting a Program or a Control Engine into the Tag File

You can insert either a control engine or a STEP 7 program into the tag file:

- Inserting a program with symbolic addressing allows you to access the symbols for the control engine (STEP 7 “station”). You can have several stations (control engines) and programs in the tag file.
- Inserting a control engine without symbolic addressing into the tag file provides the Data control with the means for accessing multiple control engines; however, you must use absolute addresses when you access the data in these control engines. Inserting a control engine does not load any symbolic information into the tag file.

Use the following procedure to create a tag file with symbolic data:

1. Select the **Insert ► Program** menu command to display the SIMATIC Program(s) browser. See Figure 9-6.
2. Click on the “+” to open the project, station, CPU, and program to be inserted in the tag file. See Figure 9-6.
3. Click on the “– →” button (or double-click on the program) to add the program to the tag file.
4. Click on the “OK” to complete the operation.

After you have inserted the program into the tag file, you can edit the “Control Engine” and “Computer Name” fields. If you change your symbol table in STEP 7, you can also update your tag file by using the the **Control Engine ► Update** menu command. The configurator uses the program path to update the symbols for the control engine.

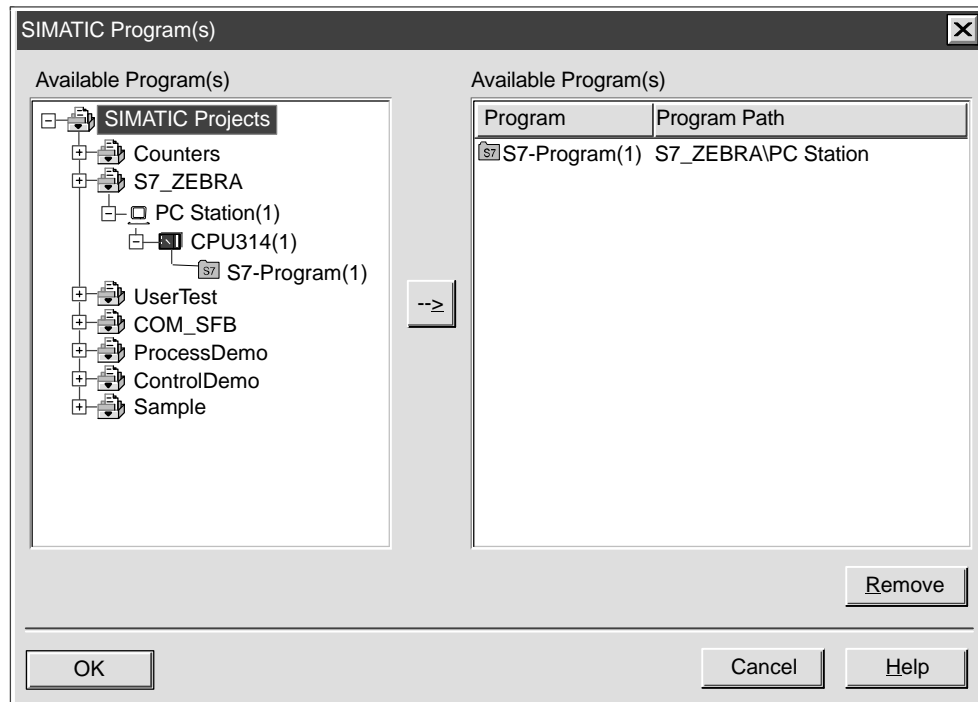


Figure 9-6 Inserting a SIMATIC Program into the Tag File

Inserting a Control Engine without STEP 7 Symbols into the Tag File

STEP 7 does not have to be installed for you to insert a control engine without symbolic information into the tag file. You must use absolute addresses when accessing data in these control engines.

Note

Do not use these characters in the Control Engine name field: “ / , \ ‘ ’ ” These characters are invalid and are not supported by this software.

Use the following procedure to insert a control engine without STEP 7 symbols:

1. Select the **Insert ► Control Engine** menu command to display the “Control Engine Configuration” dialog box. See Figure 9-7.
2. Enter the name of the computer where the control engine resides.

If the control engine is to be accessed over a local area network (DCOM), enter the DCOM address for the computer in the “Computer Name” field. Otherwise, use the default address of the local computer (“<local>”). See Section 9.4 for information about configuring the control engine for local or remote access.

3. Enter the control engine to be accessed. For example, `winLC` (for WinLC), `wcs7=3` (for a slot PLC such as the CPU 416-2 DP ISA), or `wcs7=xx,a,b` (for other PLCs on the network, where `xx` is the node address of the PLC, `a` is the rack number, and `b` is the slot number). See Appendix G for more information about control engine identifiers.
4. Enter a symbolic name for the control engine. The name defaults to the computer name plus the control engine identifier, for example `<local>_WinLC`.
5. Click on the “OK” button to enter the control engine into the tag file.

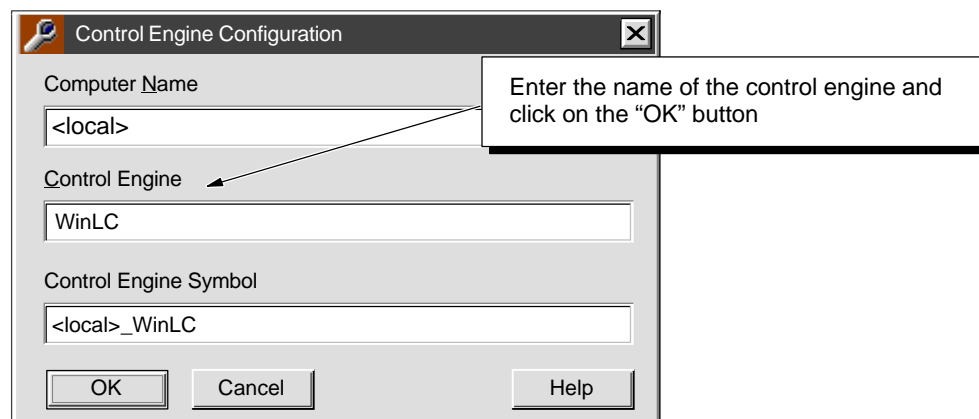


Figure 9-7 Inserting a New Control Engine into the Tag File

Using Absolute Addresses with a Tag File

You can also use absolute addresses with a tag file. Absolute addresses access the “default” control engine in the tag file. Use the following procedure to designate a default control engine:

1. Select a control engine in the file hierarchy of the tag file browser.
2. Designate the selected control engine as the default control engine by selecting the **Control Engine ► Set as Default** menu command.
3. Confirm this choice by clicking on the “Yes” button.

If no control engine in the tag file was designated as the default control engine, any absolute addresses configured in the Data control access the first control engine in the file.

9.4 Configuring a Tag File for Local or Remote Access to a Control Engine

When you insert a control engine into a tag file, you must configure the control engine as being located either on a local computer (the same computer as the Computing application and the tag file) or on a remote computer (to be accessed through DCOM).

When you use the symbols from the STEP 7 program for the control engine, the TagFile Configurator uses the following information from STEP 7 to designate the control engine:

- **Computer name:** When you use the TagFile Configurator to insert a control engine into the tag file, the default location for the control engine is the “local” computer (which designates that the control engine is located on the same computer that the Computing application and the tag file are located).

You can create a station name when you are creating your project with STEP 7 that will be interpreted by the TagFile Configurator as being a remote computer (which then designates the control engine as being located on a different computer from the Computing application and tag file). To designate a control engine as being located on a remote computer, create a station name that begins with an “@” (such as @PC 2 or @PC 3, using the sample configuration shown in Figure 9-1). The TagFile Configurator recognizes a station name in STEP 7 that begins with “@” as being a remote computer.

- **Control engine:** The TagFile Configurator reads the type of control engine from the STEP 7 project (such as WinLC, a slot PLC such as the CPU 416-2 DP ISA, or other S7 CPU). Based on the type of CPU which was configured in STEP 7, the TagFile Configurator creates an identifier for the control engine in the tag file. See Appendix G for the identifiers for the different types of control engines control engines.
- **Control engine symbol:** The TagFile Configurator constructs a symbol for the control engine by combining the following elements, separated by an underscore (“_”):
 - Computer name (the entry in the “Computer Name” field of the “Control Engine Configuration” dialog box)
 - STEP 7 symbol (symbolic name for the control engine that was created in the symbol table)

A symbol name for the control engine can now be changed. See Section 9.5.

Configuring a Control Engine for Local Access

If the control engine is running on the same computer as the Computing application, you must configure the control engine in the tag file to access the local computer:

Note

Do not use these characters in the Control Engine name field: “ / , \ ‘ ’ These characters are invalid and are not supported by this software.

1. Open the tag file and select the control engine. (For information about creating or opening a tag file, see Section 9.3.)
2. Select the **Control Engine ► Edit** menu command to display the “Control Engine Configuration” dialog box.
3. As shown in Figure 9-8, enter <local> or clear the field in the “Computer Name” field. If you leave the field empty, <local> is inserted automatically.
4. Click on the “OK” button to configure the control engine for local access.

When the Computing application uses the tag file to access the control engine, it connects to the control engine on the local computer.

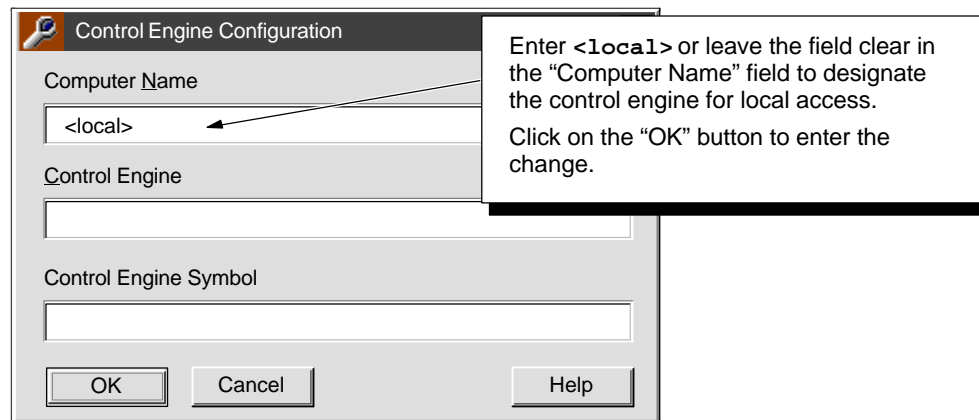


Figure 9-8 Configuring a Control Engine for Local Access

Configuring a Control Engine for Remote Access

If the control engine is running on a different computer from the Computing application, you must configure the control engine in the tag file to access the remote computer:

1. Open the tag file and select the control engine. (For information about creating or opening a tag file, see Section 9.3.)
2. Select the **Control Engine ► Edit** menu command to display the “Control Engine Configuration” dialog box.
3. As shown in Figure 9-9, enter the name of the remote control engine in the “Computer Name” field. For example, the name of the remote control engine shown in Figure 9-1 could be either **PC 2** or **PC 3**.
4. Click on the “OK” button to configure the control engine for remote access.

When the Computing application uses the tag file to access the control engine, it opens a connection through the DCOM network to the control engine that you specified.

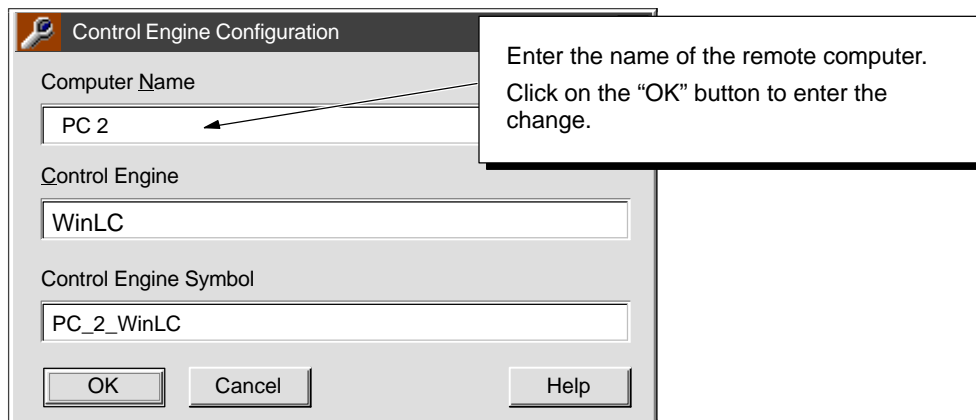


Figure 9-9 Configuring a Control Engine for Remote Access

9.5 Changing the Control Engine Symbol Name in the Tag File Editor

You can change the symbol name in the tag file editor:

1. Open the tag file and select the control engine. (For information about creating or opening a tag file, see Section 9.3.)
2. Select the **Control Engine ► Edit** menu command to display the “Control Engine Configuration” dialog box.
3. As shown in Figure 9-10, enter the name of the control engine symbol in the “Control Engine Symbol” field. For example, the name of the control engine symbol shown in Figure 9-10 could be changed to PC2WinLC.
4. Click on the “OK” button to change the control engine symbol name.

To rename the control engine, change the name in the Control Engine Symbol field. To reassign the symbol, change the name in the Control Engine field.

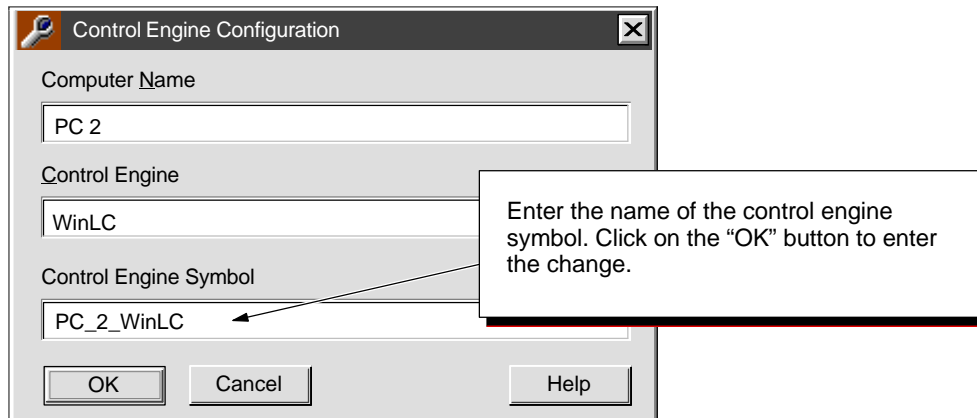


Figure 9-10 Configuring a Control Engine for Remote Access

Memory Areas of the S7 Controllers

Overview

Computing provides access to the process data within the control engine, such as an S7 programmable logic controller (PLC). Using the SIMATIC Data Control, you identify the memory area to be accessed. For more information about the memory areas, refer to the *System Software for S7-300 and S7-400 Program Design Programming Manual* or to the online help for STEP 7.

Note

Computing does not allow you to write to timers.

Section	Description	Page
A.1	Memory Areas of S7 Controllers	A-2
A.2	Accessing the S7 Data Types	A-3
A.3	Descriptions of the S7 Data Types	A-6

A.1 Memory Areas of S7 Controllers

Table A-1 lists the memory areas (including both the International and SIMATIC mnemonics) of the S7 controllers that can be accessed. Remember the following rules for accessing the peripheral I/O (PI and PQ memory areas):

- PI and PQ memory areas can be accessed only in bytes (or larger, depending on the device). You cannot access these areas as individual bits.
- The peripheral input (PI) memory area overwrites the input (I) memory area at the beginning of every scan cycle. If you use an ActiveX control to modify a value in the I memory area which has a configured peripheral, that value remains changed only until the beginning of the next scan, when the value stored in the PI memory area overwrites the changed value.

Table A-1 Memory Areas of the S7 controllers

Memory Area	Description
Peripheral input PI (International) PE (SIMATIC)	This memory area overwrites the process-image input memory at the beginning of every scan. You can access PI memory only as a byte, not as a bit.
Peripheral output PQ (International) PA (SIMATIC)	This memory area is overwritten by the process-image output area at the end of every scan. You can access the peripheral outputs only as a byte, not as a bit.
Process-image input I (International) E (SIMATIC)	This memory area is overwritten by the peripheral input memory area at the beginning of every scan.
Process-image output Q (International) A (SIMATIC)	This memory area overwrites the peripheral output memory area at the end of every scan.
Bit memory M (International and SIMATIC)	This memory area provides storage for interim results calculated in the program.
Timer T (International and SIMATIC)	This memory area provides the timers used by the program. Computing allows you to only read timers. You cannot write data to timers.
Counter C (International) Z (SIMATIC)	This memory area provides the counters used by the program.
Data block DB (International and SIMATIC)	The DB memory address references the data stored in the data blocks for the program.

A.2 Accessing the S7 Data Types

You access data in the control engine by assigning a property in the object to a variable (memory location) in the control engine. As shown in Figure A-1, the most significant byte for the data is the byte of the address. For example, the most significant byte for MW0 is byte 0, and the most significant byte for MD0 is byte 0. Table A-2 lists the valid data types that you can enter when assigning variables.

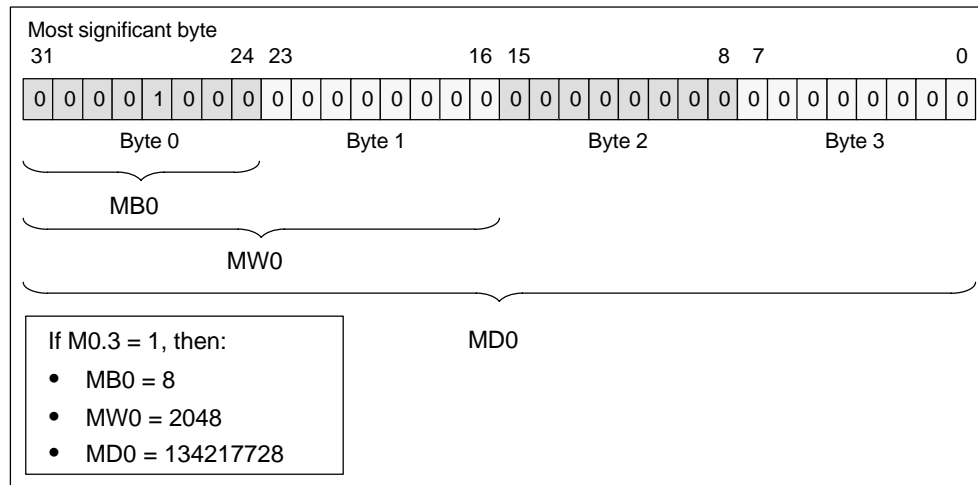


Figure A-1 Accessing Data by Byte, Word, and Double Word

Computing also allows you to specify a data type when you assign a variable to one of the properties of a control. You define the data type by entering the absolute address for the memory location, followed by a colon (:) and the data type. For example, you can define an assigned variable as a REAL data type by entering “MD100:real” when you assign the variable. If you do not specify a data type, Computing uses the default data types listed in Table A-2.

You can also access data stored in arrays or strings. For example, to access the second value in a one-dimensional array of REAL data, enter “MD100:real[2]”.

Note

You can access most of the S7 data types from other applications without having to provide any external interpretation of the data. These include BOOL, BYTE, CHAR (character), WORD, DWORD, INT (integer), DINT (double integer), and REAL (floating point). Some of the S7 data types are specific to SIMATIC products: DATE, S5TIME, TIME, TIME_OF_DAY (TOD), and DATE_AND_TIME.

The SIMATIC Number control automatically transforms these data types; however, if you access these S7-specific data types with other controls, you must also manually transform the data. Refer to the descriptions of these data types that follow Table A-2.

Table A-2 Addressing the S7 Data Types and S7 Memory Areas

Memory Area	Address	Valid Data Type
Peripheral Output	PABx (SIMATIC) PQBx (International)	BYTE (default), CHAR
	PAWx (SIMATIC) PQWx (International)	WORD (default), INT, DATE, S5TIME
	PADx (SIMATIC) PQDx (International)	DWORD (default), DINT, REAL, TOD, TIME
Peripheral Input	PEBx (SIMATIC) PIBx (International)	BYTE (default), CHAR
	PEWx (SIMATIC) PIWx (International)	WORD (default), INT, DATE, S5TIME
	PEDx (SIMATIC) PIDx (International)	DWORD (default), DINT, REAL, TOD, TIME
Output	Ax.y (SIMATIC) Qx.y (International)	BOOL (default)
	ABx (SIMATIC) QBx (International)	BYTE (default), CHAR
	AWx (SIMATIC) QWx (International)	WORD (default), INT, DATE, S5TIME
	ADx (SIMATIC) QDx (International)	DWORD (default), DINT, REAL, TOD, TIME
Input	Ex.y (SIMATIC) Ix.y (International)	BOOL (default)
	EBx (SIMATIC) IBx (International)	BYTE (default), CHAR
	EWx (SIMATIC) IWx (International)	WORD (default), INT, DATE, S5TIME
	EDx (SIMATIC) IDx (International)	DWORD (default), DINT, REAL, TOD, TIME
Bit Memory	Mx.y	BOOL (default)
	MBx	BYTE (default), CHAR
	MWx	WORD (default), INT, DATE, S5TIME
	MDx	DWORD (default), DINT, REAL, TOD, TIME
Data Block (DB)	DBz.DBx.y DBz.DBXx.y	BOOL (default)
	DBz.DBBx	BYTE (default), CHAR
	DBz.DBWx	WORD (default), INT, DATE, S5TIME
	DBz.DBDx	DWORD (default), DINT, REAL, TOD, TIME

Table A-2 Addressing the S7 Data Types and S7 Memory Areas, continued

Memory Area	Address	Valid Data Type
Timer (read-only)	Tx	INT (default)
Counter	Zx (SIMATIC) Cx (International)	INT (default)

Note

Computing does not allow you to write to timers.

Table A-3 lists the S7 data types and the corresponding data types for C and Visual Basic.

Table A-3 S7 Data Types as C or Visual Basic Data Types

S7 Data Type	C Data Type	VB Data Type
ARRAY	VT_ARRAY	Not applicable
BOOL (Boolean)	VT_BOOL	Boolean
BYTE	VT_UI1	Byte
CHAR (character)	VT_BSTR	String
DATE	VT_DATE	Date
DATE_AND_TIME	VT_DATE	Date
DINT (double integer)	VT_I4	Long
DT (date and time)	VT_DATE	Date
DWORD (double word)	VT_CY ¹	Currency ¹
INT (integer)	VT_I2	Integer
REAL	VT_R4	Single
S5TIME	VT_I4	Long
STRING	VT_BSTR	String
TIME	VT_I4	Long
TOD (time of day)	VT_DATE	Date
WORD	VT_I4	Long

¹ If you are reading DWORD data to Excel, you must reformat the data type for the Excel file or cell from "General" to the "Number" data type. Otherwise, Excel formats the S7 DWORD data into a monetary value, using the "Currency" data type for Excel.



Caution

Using the timer function improperly or using breakpoints in Visual Basic with Computing can cause problems that could potentially cause your computer or application to crash or lock up. Depending on the configuration, this could cause the application to lose communication with the control engine. This could cause unpredictable process operation, which could result in death or serious injury to personnel, and/or damage to equipment.

Always install a physical emergency stop circuit for your machine or process.

Concerning VB timers: The Timer function in Visual Basic version 5 allows a timer to interrupt code in progress within the same thread, which can cause problems with potentially serious consequences. If you use VB timers with Computing, observe the following guidelines:

- Always kill (disable) the timers in the Form_Unload subroutine. Otherwise, a timer can trigger an event while the VB program is shutting down; this condition could cause your computer or your application to crash, lock up, or to continue running invisibly.
 - If you start your timer in the Form_Load subroutine, the timer event could occur before the other objects have been instantiated. In order to ensure that the objects have been properly instantiated, always start a timer in the Form_Load subroutine with a large interval (such as 1 or 2 seconds) to allow the objects to be properly instantiated. Subsequent timer intervals can be set to shorter intervals.
-

A.3 Descriptions of the S7 Data Types

Accessing Data in an ARRAY

Each dimension of an array of bits, bytes, or characters is aligned on byte boundaries; for all other arrays, each dimension is aligned on word boundaries. Figure A-2 shows how a sample array is stored in memory. The operating system calculates the bit address for the end position of each element in the array. The array is then filled to the next word (or byte) address; the next data type starts on the next word (or byte) boundary.

Multi-dimensional array are stored sequentially For the example shown in Figure A-2, integer [1,1] is followed by integer [1,2], and integer [1,3] is followed by integer [2,1].

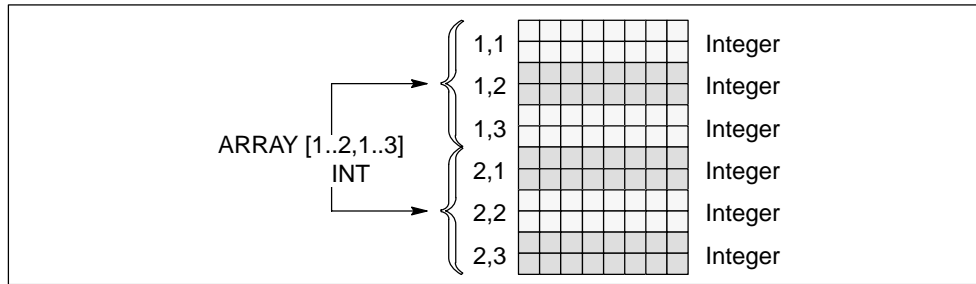


Figure A-2 Accessing Data in an ARRAY

To read an array from the control engine, use the `ReadVariable` property of the Data control. For example, the following code reads a one-dimensional array of integers (starting at MW0) into an array named "MyVariant":

```
S7Data1.ReadVariable("MW0:INT[100]", MyVariant, MyState, MyTimeout)
```

Use the following code reads a one-dimensional array of 100 bits (starting with M0.0) into an array named "MyVariant":

```
S7Data1.ReadVariable("M0.0:[100]", MyVariant, MyState, MyTimeout)
```

Refer to Section 5.11 for a sample program that reads and writes arrays.

Accessing the DATE Data Type

The DATE data type is stored as a positive integer which represents the number of days since January 1, 1990. See Figure A-3. The valid dates range from January 1, 1990 to December 31, 2168.

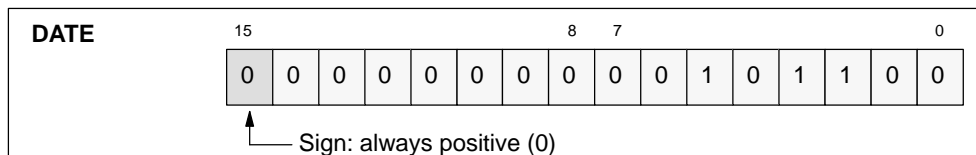


Figure A-3 Accessing the DATE Data Type

Accessing the DATE_AND_TIME Data Type

The DATE_AND_TIME data type is stored as an 8 byte variant with the format shown in Figure A-4. The range for each byte is as follows:

- Year: 1990 to 1999, 2000 to 2089 (BCD: 90h to 99h, 90h to 99h)???
- Month: 1 to 12 (BCD: h to 12h)
- Day: 1 to 31 (BCD: h to 31h)
- Hour: 00 to 23 (BCD: 00h to 23h)
- Minute: 00 to 59 (BCD: 00h to 59h)
- Second: 00 to 59 (BCD: 00h to 59h)
- Millisecond: 0 to 999 (BCD: 000h to 999h)
- Day of Week: Sunday (1) to Saturday (7) (BCD: h to 7h)

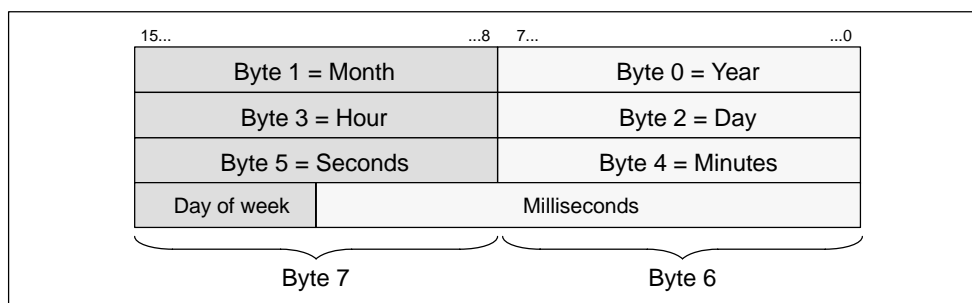


Figure A-4 Accessing the DATE_AND_TIME Data Type

Accessing the S5TIME Data Type

In the S7 controller, the S5TIME data type provides information such as time base and value (stored in BCD format). However, when you use Computing to read S5TIME data, Computing returns a VB Long variable that contains the timer value in milliseconds (ms).

Accessing STRING Data

A STRING is a grouping (or “string”) of ASCII characters that can be up to 254 characters. Each character in the string is stored in one byte. You specify the number of characters in the string to be accessed. For example: STRING[25] accesses a string that contains 25 characters.

Note

Read and write STRING or CHAR data as a Visual Basic BSTR data. Do not use an array of CHAR to emulate a STRING.

Use one BSTR for each STRING or CHAR, regardless of the length of the data being accessed. For example:

- To access CHAR[50] (which designates 50 bytes or 50 characters), use a BSTR of up to 50 bytes and **not** 50 individual BSTRs.
- To access STRING[50] (which designates a string of 50 characters), use a BSTR of up to 50 bytes and **not** 50 individual BSTRs.

As shown in Figure A-5, the memory that is allocated for a string includes a header (2 bytes) which include the following information:

- The first byte stores the maximum length of memory for the string (the default value is 256 bytes).
- The second byte stores the actual amount of memory used for the string.

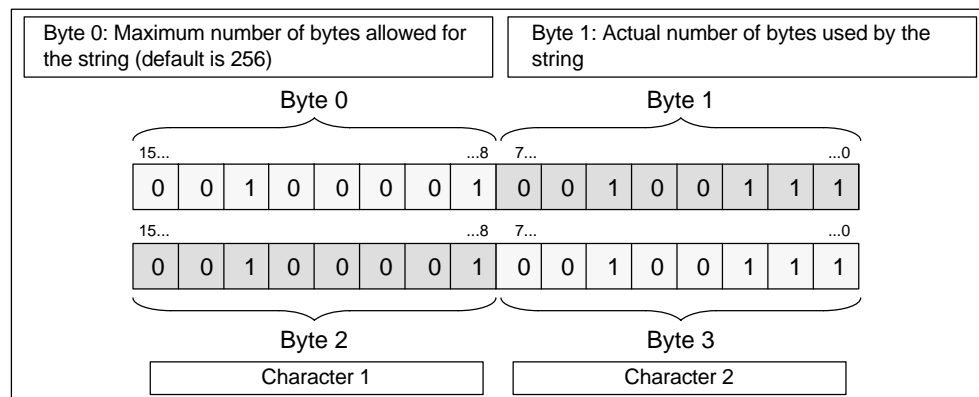


Figure A-5 Accessing STRING Data

Accessing the TIME Data Types

The TIME data type is stored as a signed, double integer for the number of milliseconds (ms), ranging from –24 days, 20 hours, 31 minutes, 23 seconds, and 648 ms to +24 days, 20 hours, 31 minutes, 23 seconds, and 647 ms.

Negative values are represented as the two's complement of the positive number. (To create the two's complement of the data, invert the signal states of all bits and then add + 1 to the result.)

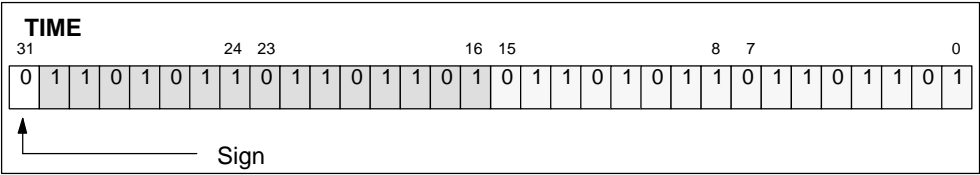


Figure A-6 Accessing the TIME Data Type

Accessing the TIME_OF_DAY (TOD) Data Type

The TIME_OF_DAY (TOD) data type is stored as a positive double integer for the number of milliseconds (ms) since midnight, ranging from 0:0:0.0 to 23:59:59.999.

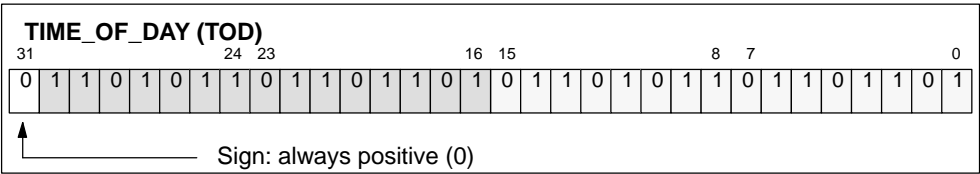


Figure A-7 Accessing the TIME_OF_DAY (TOD) Data Type

Properties and Methods *in work*

B.1 AboutBox Method

Applies to: Button, Edit, Label, Slider

This method displays the “About” message box for the control.

Syntax:

object.**AboutBox**

The AboutBox method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.

B.2 Activated Property

Applies to: Data

This property allows you to specify whether or not all connections are activated.

Syntax:

object.**Activated** [= *value*]

The Activated property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> can respond to user-generated events.

The settings for *value* are:

Setting	Description
True	(default) All connections are activated.
False	All connections are deactivated.

Note

The connections remain established, even if they are deactivated.

B.3 Alignment Property

Applies to: Button, Edit, Label

This property specifies the alignment of the text of the control.

Syntax:

object.Alignment [= *value*]

The Alignment property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the alignment.

The settings for *value* are:

Setting	Description
0 or Left	(Default for Edit Control) Left-aligned.
1 or Right	Right-aligned.
2 or Center	(Default for Button and Label Controls) Centered.

B.4 Appearance Property

Applies to: Button, Edit, Label

If this property is set to ThreeD (1) and the BorderStyle property is set to “Fixed Single” (1), then the Appearance property draws controls with three-dimensional effects. If the property is set to Flat (0), a flat border will surround the controls rectangle.

Note

This property only has an effect if the BorderStyle property is set to “Fixed Single” (1).

Syntax:

object.**Appearance** [= *value*]

The Appearance property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A <i>value</i> or constant that determines the appearance of <i>object</i> .

The settings for *value* are:

Setting	Description
0 or Flat	Paints the controls and forms without visual effects.
1 or ThreeD	(Default) Paints the controls with three-dimensional (3-D) effects.

B.5 AutoConnect Property

Applies to: Data

This property allows you to specify whether or not the configured connections are established at runtime.

Syntax:

object.**AutoConnect** [= *value*]

The AutoConnect property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A boolean expression that specifies whether <i>object</i> can respond to user-generated events.

The settings for *value* are:

Setting	Description
True	(default) All configured connections will be established at runtime.
False	The connections will be established with a call to the Connect method.

Note

If you explicitly call the Connect method within your program, disable the AutoConnect property for the Data control. This helps to ensure that the Data control does not connect unexpectedly to the control engine.

B.6 AutoConnectTimeout Property

Applies to: Data

This property allows you to specify a timeout. After the time specified, the Data control issues a call to its Connect method if the AutoConnect property is set to True. The value can also be specified at the General Property Tab.

Syntax:

object.AutoConnectTimeout [= *value*]

The AutoConnectTimeout property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value of the type Long, which states the timeout in milliseconds.

B.7 BackColor Property

Applies to: Edit, Label, Slider

This property returns or sets the background color of the control.

Syntax:

object.BackColor [= *color*]

The BackColor property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>color</i>	A value or constant that determines the background color of an object.

The settings for *color* are:

Setting	Description
Standard Colors	Colors specified by using the RGB Color palette
Windows System Colors	Colors specified by system color constants (depending on the container); for example, colors listed in the Visual Basic (VB) object library in the Object Browser

B.8 bDiagBuffOK Property

Applies to: DBuffer

This read-only property verifies whether there is a connection to the diagnostics buffer of the S7 control engine.

Syntax:

object.bDiagBuffOK [= *value*]

The bDiagBuffOK property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether there is a connection to the diagnostic buffer of the S7 control engine.

The settings for *value* are:

Setting	Description
True	The connection to the diagnostic buffer has been verified and is active.
False	(default) There is no connection to the diagnostic buffer.

B.9 bEngineConnected Property

Applies to: DBuffer

This read-only property verifies that the control has established a connection with an S7 control engine.

Syntax:

object.bEngineConnected [= *value*]

The bEngineConnected property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> is connected to an S7 control engine.

The settings for *value* are:

Setting	Description
True	The connection to the S7 control engine has been verified and is active.
False	(default) There is no connection to the S7 control engine.

B.10 BorderStyle Property

Applies to: Edit, Button, Label

If the property has the value "1-Fixed Single", the control is surrounded by a rectangular border. If the property has the value "0-wNone", no border will be displayed.

Note

This property determines whether the Appearance property has any effect.

Syntax:

object.BorderStyle [= *value*]

The BorderStyle property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the border style.

The settings for *value* are:

Setting	Description
0 or None	(Default) No border or border-related elements
1 or FixedSingle	A fixed, single-line border

B.11 Caption Property

Applies to: Label

This property specifies the text that is displayed by the control.

Syntax:

object.Caption [= *value*]

The Caption property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A String value that specifies the text of the label.

B.12 Connect Method

Applies to: Data

This method establishes all configured connections.

Note

If your subroutine accesses the Data control programmatically, always disconnect from the control engine (using a Disconnect method) in the Form_Unload subroutine.

In addition, disable the AutoConnect property for the Data control if you explicitly call the Connect method within your program. This helps to ensure that the Data control does not connect unexpectedly to the control engine.

Syntax:

result = **object.Connect**

The Connect method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.

B.13 ConnectName Method

Applies to: Data

This method establishes connections for the object that is specified by the name of the object on the form.

If your subroutine accesses the Data control programmatically, always disconnect from the control engine (using a Disconnect method) in the Form_Unload subroutine.

In addition, disable the AutoConnect property for the Data control if you explicitly call the Connect method within your program. This helps to ensure that the Data control does not connect unexpectedly to the control engine.

Note

A programmer who uses Visual Basic (or a similar programming language) would use the ConnectName method, while a programmer who uses Visual C (or a similar programming language) would use the ConnectObject method.

Syntax:

result = **object.ConnectName** *ConnectedObject*, *ConnectionTable*

The ConnectName method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.
<i>ConnectedObject</i>	A String expression that evaluates to the name of an object that should be connected. If this parameter is set to an empty String, the control generates the ValueChanged event if a connected variable changes.
<i>ConnectionTable</i>	(optional) Specifies a connection table. If this parameter is omitted, the control reads the ConnectionTable property of the ConnectedObject. The connection table is declared as an array. Each element in the array has the following parts: <ul style="list-style-type: none">• Name of the element (such as "Value")• Memory location (such as MW100)• Update rate or time-out value (in ms)• Deadband value For more information about the connection table, see Section 5.7.

Note

If the ConnectedObject and ConnectionTable parameters are both omitted, an error is reported.

B.14 ConnectObject Method

Applies to: Data

This method establishes connections for a specified object which was declared in the program.

If your subroutine accesses the Data control programmatically, always disconnect from the control engine (using a Disconnect method) in the Form_Unload subroutine.

In addition, disable the AutoConnect property for the Data control if you explicitly call the Connect method within your program. This helps to ensure that the Data control does not connect unexpectedly to the control engine.

Note

A programmer who uses Visual Basic (or a similar programming language) would use the ConnectName method, while a programmer who uses Visual C (or a similar programming language) would use the ConnectObject method.

Syntax:

result = *object*.ConnectObject *ConnectedObject*, *ConnectionTable*

The ConnectObject method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.
<i>ConnectedObject</i>	A String expression that evaluates to the name of an object that should be connected. If this parameter is set to an empty String, the control generates the ValueChanged event if a connected variable changes.
<i>ConnectionTable</i>	(optional) Specifies a connection table. If this parameter is omitted, the control reads the ConnectionTable property of the ConnectedObject. The connection table is declared as an array. Each element in the array has the following parts: <ul style="list-style-type: none"> • Name of the element (such as "Value") • Memory location (such as MW100) • Update rate or time-out value • Deadband value

Note

If the ConnectedObject and ConnectionTable parameters are both omitted, an error is reported.

B.15 ControlEngine Property

Applies to: Data, DBuffer

This property stores the pathname or identification of the control engine connected to the control. See Appendix G for more information about control engine strings.

Syntax:

`object.ControlEngine [= value]`

The ControlEngine property has these parts:

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

B.16 DataFormat Property

Applies to: Edit

This property defines the storage type used for converted values. If you are using a data format for displaying a value which is too large, the value will be truncated.

Note

This property determines whether the Precision property has any effect.

Syntax:

`object.DataFormat [= value]`

The DataFormat property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the data format, as described in Table B-1.

Table B-1 Settings for the Data Format

Constant	Setting	Description
wBoolean	0	Bit value
wBinary	1	Any bit, byte, word, dword, int or dint value
wOctal	2	Any byte, word, dword, int or dint value
wHexadecimal	3	Any byte, word, dword, int or dint value
wUnsignedDecimal	4	Any byte, word, dword, int or dint value
wSignedDecimal	5	Any byte, word, dword, int or dint value

Table B-1 Settings for the Data Format

Constant	Setting	Description
wReal	6	4-byte floating point value
wTimer	7	2-byte signed integer value
wCounter	8	2-byte signed integer value
wTime	9	Signed integer value (IEC Time)
wDate	10	Signed integer value (IEC Date)
wTimeOfDay	11	Signed integer value (IEC Time)
wChar	12	1-byte ASCII character
wString	13	String of characters

Note

If the data size configured to be accessed in the control engine is larger than the data being displayed in the Edit control and the value of the data from the PLC is larger than can be displayed by the data format, the value is displayed with “...” preceding it. Before the value can be changed from the Edit Control, the “...” preceding the value must be deleted.

When a value is written from the Edit Control to the PLC with this configuration, the amount of data written to the PLC corresponds to the data size configured in the Data Control. Therefore, caution must be taken that memory locations are not changed inadvertently.

B.17 DefaultDeadband Property

Applies to: Data

This property allows you to specify the dead band used by the Data control, if no dead band is specified in the connection table.

Note

If you specify a dead band (such as “10”) for a bit variable (such as M15.5), the control engine will not transmit a changed value for that bit.

Syntax:

`object.DefaultDeadBand [= value]`

The DefaultDeadband property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value of the type Single, which must not be negative.

B.18 DefaultUpdateRate Property

Applies to: Data

This property allows you to specify the update rate used by the Data control, if no update rate is specified in the connection table.

Note

For WinLC, the minimum default update rate is 0. For CPU 416-2 DP ISA, the minimum default update rate is 100 ms.

Syntax:

`object.DefaultUpdateRate [= value]`

The DefaultUpdateRate property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value of type Long.

The settings for *value* are:

Part	Description
0	All changes of the connected variable are reported immediately.
> 0	Changes of the connected variable are reported after this timeout.

B.19 Direction Property

Applies to: Slider

This property sets the orientation (horizontal or vertical) of the SIMATIC control. Default is 0 – wHorizontal.

Syntax:

object.Direction [= *value*]

The Direction property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the orientation.

The settings for *value* are:

Setting	Description
0	(Default) wHorizontal
1	wVertical

B.20 Disconnect Method

Applies to: Data

This method releases all established connections.

Note

If your subroutine accesses the Data control programmatically, always disconnect from the control engine (using a Disconnect method) in the Form_Unload subroutine.

In addition, disable the AutoConnect property for the Data control if you explicitly call the Connect method within your program. This helps to ensure that the Data control does not connect unexpectedly to the control engine.

Syntax:

result = *object.Disconnect*

The Disconnect method has these parts:

Part	Description
<i>object</i>	An object expression (identifier for the specific SIMATIC control) that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.

B.21 DisplayFormatButtons Property

Applies to: DBuffer

This property allows you to show or hide the format buttons (Text or Hexadecimal).

Syntax:

`object.DisplayFormatButtons [= value]`

The DisplayFormatButtons property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> displays the buttons.

The settings for *value* are:

Setting	Description
True	(default) The Text and Hexadecimal (Hex) buttons are displayed.
False	The Text and Hexadecimal (Hex) buttons are not displayed.

B.22 DisplayHelpButton Property

Applies to: DBuffer

This property allows you to show or hide the button for displaying the online help for the control.

Syntax:

`object.DisplayHelpButton [= value]`

The DisplayHelpButton property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> displays the button.

The settings for *value* are:

Setting	Description
True	(default) The button for the online help of <i>object</i> is displayed.
False	The button is not displayed.

B.23 DisplayHelpOnEventButton Property

Applies to: DBuffer

This property allows you to show or hide the button for displaying the online help for the selected diagnostic event.

Syntax:

`object.DisplayHelpOnEventButton [= value]`

The DisplayHelpOnEventButton property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> displays the button.

The settings for *value* are:

Setting	Description
True	(default) <i>object</i> displays the button for the online help of the selected diagnostic event.
False	The button is not displayed.

B.24 DisplayLowerPanel Property

Applies to: DBuffer

This property allows you to show or hide the lower panel of the diagnostics buffer.

Syntax:

`object.DisplayLowerPanel [= value]`

The DisplayLowerPanel property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> displays the button.

The settings for *value* are:

Setting	Description
True	(default) <i>object</i> displays the lower panel of the diagnostic buffer.
False	The lower panel is not displayed.

B.25 DisplayUpdateButton Property

Applies to: DBuffer

This property allows you to show or hide the button for updating the control by reading the entries of the diagnostics buffer of the control engine. The control reads the diagnostics buffer only when an update is requested.

Syntax:

`object.DisplayUpdateButton [= value]`

The DisplayUpdateButton property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> displays the button.

The settings for *value* are:

Setting	Description
True	(default) <i>object</i> displays the button for updating the diagnostic events.
False	The button is not displayed.

B.26 DisplayUpperPanel Property

Applies to: DBuffer

This property allows you to show or hide the upper panel of the diagnostics buffer.

Syntax:

`object.DisplayUpperPanel [= value]`

The DisplayUpperPanel property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> displays the button.

The settings for *value* are:

Setting	Description
True	(default) <i>object</i> displays the upper panel of the diagnostics buffer.
False	The upper panel is not displayed.

B.27 DisplayValue Property

Applies to: Edit, Slider

This property is a variant which returns the scaled value for the control.

Syntax:

`object.DisplayValue [= value]`

The DisplayValue property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Variant that specifies the value of the control.

B.28 Enabled Property

Applies to: Button, Edit, Label, Slider

When this property is True, the control reacts on changes of the Value property and fires events. If this property is False, then the control is disabled and does not react on changes in the Value property and does not fire any event (except the error event).

Syntax:

object.Enabled [= *boolean*]

The Enabled property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>boolean</i>	A boolean expression that specifies whether <i>object</i> can respond to user-generated events.

The settings for *boolean* are:

Setting	Description
True	(Default) Allows the object to respond to events
False	Prevents object from responding to events

B.29 EnableSort Property

Applies to: DBuffer

This property allows you to determine whether the user can cause the entries to be sorted. When this property is enabled, the user clicks on a column heading and the entries are sorted according to that column.

Syntax:

`object.EnableSort [= value]`

The EnableSort property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that enables or disables the sorting of the columns in the upper panel of the diagnostics buffer.

The settings for *value* are:

Setting	Description
True	(default) The entries in the upper panel of the diagnostics buffer are sorted when the user clicks on a column heading.
False	The sorting of entries is disabled.

B.30 Factor Property

Applies to: Edit, Slider

The Factor and Offset properties specify the scaling factor and the offset used when the scale-by-formula option has been enabled.

Note

The ScaleMode property must be set to “wByFormula” (1) for the Factor and Offset properties to have any effect.

You can use a formula to scale the value. In the following formula, “Value” is similar to the contents of the Value property if the control is connected to the control engine; “Factor” is the value of the Factor property; “Offset” is the value of the Offset property; and “DisplayValue” is also the contents of the Text property.

$\text{Value} * \text{Factor} + \text{Offset} = \text{DisplayValue}$

Syntax:

`object.Factor [= value]`

The Factor property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A floating-point value that defines the factor for the scaling formula.

Note

The default value of the factor is 1.0, and the default value of the offset is 0.0.

B.31 FalseCaption Property

Applies to: Button

This property determines the text that is displayed in the control when the Value property is False (equal to 0, or “Off”).

Syntax:

`object.FalseCaption [= string]`

The FalseCaption property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>string</i>	Text that determines the active or inactive text of the control

B.32 FalseColor Property

Applies to: Button

This property determines the color of the control when the Value property is False (equal to 0, or “Off”).

Syntax:

`object.FalseColor [= color]`

The FalseColor property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>color</i>	A value or constant that determines the background or foreground colors of an object.

The settings for *value* are:

Setting	Description
Standard Colors	Colors specified by using the RGB Color palette
Windows System Colors	Colors specified by system color constants (depending on the container); for example, colors listed in the Visual Basic (VB) object library in the Object Browser

B.33 FalsePicture Property

Applies to: Button

This property returns or sets the inactive (off, false, etc.) picture displayed on the control.

Syntax:

object.FalsePicture [= *picture*]

The FalsePicture property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>picture</i>	A picture that determines the image of an object.

B.34 Font Property

Applies to: Button, Edit, Label

This property returns a Font object for the main font of the control.

Syntax:

object.Font [= *font*]

The Font property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>font</i>	A value that returns or sets the font used for the control.

B.35 ForeColor Property

Applies to: Button, Edit, Label, Slider

This property returns or sets the foreground color used to display text and graphics in an object.

Syntax:

`object.ForeColor [= color]`

The ForeColor property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>color</i>	A value or constant that determines the foreground colors of <i>object</i> .

The settings for *color* are:

Setting	Description
Standard Colors	Colors specified by using the RGB Color palette
Windows System Colors	Colors specified by system color constants (depending on the container); for example, colors listed in the Visual Basic (VB) object library in the Object Browser

B.36 FormatDisplay Property

Applies to: DBuffer

This property allows you to change the format of the additional information of a selected diagnostic event.

Syntax:

`object.FormatDisplay [= value]`

The FormatDisplay property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> displays the information for the diagnostic event as text or hexadecimal values.

The settings for *value* are:

Setting	Description
True	(default) The information for the diagnostic event is displayed as text
False	The information is displayed as hexadecimal numbers

B.37 KnobHeight Property

Applies to: Slider

This property determines the height of the indicator displayed by the control.

Syntax:

object.KnobHeight [= *single*]

The KnobHeight property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>single</i>	A value that determines the height of the indicator.

B.38 KnobPicture Property

Applies to: Slider

This property determines the picture (graphic) used for the indicator on the control.

Syntax:

object.KnobPicture [= *picture*]

The KnobPicture property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>picture</i>	A picture that determines the image for the indicator.

B.39 KnobWidth Property

Applies to: Slider

This property determines the width of the indicator displayed by the control.

Syntax:

object.KnobWidth [= *single*]

The KnobWidth property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>single</i>	A value that determines the width of the indicator.

B.40 LargeChange Property

Applies to: Slider

This property determines how far the slider indicator moves when the control has focus and you press the Page Up or Page Down key. The Value property is increased by LargeChange if you press the Page Up key or click to the right of (above) the indicator. It is decreased by LargeChange if you press the Page Down key or click to the left of (below) the indicator.

Syntax:

object.LargeChange [= *value*]

The LargeChange property has these parts:

Part	Description
<i>object</i>	An object expression (identifier for the specific SIMATIC control) that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the amount of change.

B.41 Locked Property

Applies to: Button, Edit, Slider

If the control is locked it is in a read-only state. The user is unable to change any values, but the current value is still displayed. By default the control is not in locked mode, so you can enter numbers.

Syntax:

```
object.Locked [= boolean]
```

The Locked property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>boolean</i>	A Boolean expression that specifies whether the control can be edited.

The settings for *boolean* are:

Setting	Description
True	You can scroll and highlight the text in the control, but you cannot edit it. Changes to the Value property will be reflected. This means that the control still shows values in the PLC, but the user is unable to change them.
False	(Default) You can edit the text in the control.

B.42 Max and Min Properties

Applies to: Edit, Slider

If the ScaleMode property is wByRange or wScaleNone, these properties return/set the maximum/minimum scaled value of the control.

Syntax:

```
object.Max [= value]
```

```
object.Min [= value]
```

The Max and Min properties have these parts:

Part	Description
<i>object</i>	An object expression (identifier for the specific SIMATIC control) that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that specifies maximum/minimum scaled value of the control.

B.43 MultipleEngines Property

Applies to: Data

This property specifies whether the control connects to a specific control engine or connects simultaneously to several control engines. See Appendix G for more information about control engine strings.

Syntax:

`object.MultipleEngines [= value]`

The MultipleEngines property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether <i>object</i> connects to one or to several control engines.

The settings for *value* are:

Setting	Description
True	<i>object</i> connects to more than one control engine simultaneously.
False	(default) <i>object</i> connects only to the control engine specified in the ControlEngine property.

B.44 Offset Property

Applies to: Edit, Slider

The Factor and Offset properties specify the scaling factor and the offset used when the scale-by-formula option has been enabled.

Note

The ScaleMode property must be set to “wByFormula” (1) for the Factor and Offset properties to have any effect.

You can use a formula to scale the value. In the following formula, “Value” is similar to the contents of the Value property if the control is connected to the control engine; “Factor” is the value of the Factor property; “Offset” is the value of the Offset property; and “DisplayValue” is also the contents of the Text property.

$\text{Value} * \text{Factor} + \text{Offset} = \text{DisplayValue}$

Syntax:

object.Offset [= *value*]

The Offset property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A floating-point value that defines the factor or the offset for the scaling formula.

Note

The default value of the factor is 1.0, and the default value of the offset is 0.0.

B.45 PCName Property

Applies to: Data

This property selects the name of a remote computer (PC) in order to connect to a control engine over a network, such as a local area network (LAN). See Appendix G for more information about control engine strings.

Syntax:

object.PCName [= *value*]

The PCName property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A String that specifies the pathname or identification of the remote computer (PC) for the connection.

B.46 Picture Property

Applies to: Slider, Label

This property determines the picture (graphic) used for the control.

Syntax:

object.Picture [= *picture*]

The Picture property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>picture</i>	A picture that determines the image of the object.

B.47 PopUpHelp Method

Applies to: DBuffer

This method displays the online help for the S7 control.

Syntax:

result = *object*.PopUpHelp

The PropertyChangedObject method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.

B.48 PopUpHelpOnEvent Method

Applies to: DBuffer

This method displays the online help for the selected diagnostics event.

Syntax:

result = *object*.PopUpHelpOnEvent

The PropertyChangedObject method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.

B.49 Precision Property

Applies to: Edit

This property is available if the DataFormat is set to "Real" (6) (data type with precision). In that case you can change the precision (number of digits behind the decimal point) of the number. The number will be rounded at the specified precision.

Note

The DataFormat property must be set to "Real" (6) before this property can have an effect.

Syntax:

object.Precision [= *value*]

The Precision property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	An integer value that defines the precision of the number. The default precision is 3.

B.50 PropertyChangedName Method

Applies to: Data

This method notifies the Data control that the value of a property of a connected control, referenced by the name of the object in the form, has changed. The Data control reads the value from the property and writes it to the data source.

Note

A programmer who uses Visual Basic (or a similar programming language) would use the PropertyChangedName method, while a programmer who uses Visual C (or a similar programming language) would use the PropertyChangedObject method.

Syntax:

result = *object.PropertyChangedName* *ConnectedObject*, *Property*

The PropertyChangedName method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.
<i>ConnectedObject</i>	A String expression that evaluates to the name of the connected control whose property has changed.
<i>Property</i>	A String value with the name of the property that has changed.

B.51 PropertyChangedObject Method

Applies to: Data

This method notifies the Data control that the value of a property of a connected control (an object which was declared in the program) has changed. The Data control reads the value from the property and writes it to the data source.

Note

A programmer who uses Visual Basic (or a similar programming language) would use the PropertyChangedName method, while a programmer who uses Visual C (or a similar programming language) would use the PropertyChangedObject method.

Syntax:

result = *object*.**PropertyChangedObject** *ConnectedObject*, *Property*

The PropertyChangedObject method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.
<i>ConnectedObject</i>	A String expression that evaluates to the name of the connected control whose property has changed.
<i>Property</i>	A String value with the name of the property that has changed.

B.52 PushButton Property

Applies to: Button

Determines the operation mode of the control: if set to “True” or 1, the Value property is inverted as long as the Button control is “pressed” (MouseDown event)

Syntax:

```
object.PushButton [= boolean]
```

The PushButton property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>boolean</i>	An boolean expression that specifies the operation mode of the control.
Setting	Description
True	The button is pressed; the Value property is inverted.
False	(default) The button is not pressed.

B.53 RawMax and RawMin Properties

Applies to: Edit, Slider

These properties define the ranges for scaling a value:

- RawMax specifies the maximum raw value of the control if the ScaleMode is wByRange or wScaleNone.
- RawMin specifies the minimum raw value of the control if the ScaleMode is wByRange or wScaleNone.

Note

The ScaleMode property must be set to “wByRange” or “wScaleNone” before these properties can have an effect.

When you use a range transformation to scale the value, you specify a source range (for the values in the control engine) and a destination range (for the values that are displayed by the control). The values of one range will be transformed to the other range. The source and destination ranges define a ratio for the transformation; they do not define upper or lower limits. A value can be larger or smaller than the range; the transformation will use the two ranges to extrapolate the other value.

Syntax:

object.RawMax [= *value*]

object.RawMin [= *value*]

The RawMin and RawMax properties have these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the maximum or minimum raw value of the control.

B.54 ReadMultiVariables Method

Applies to: Data

This method reads the status of the connected variables in the control engine.

Syntax:

result = *object*.ReadMultiVariables (*VarNames*, *VarValues*, *States*)

The ReadMultiVariables method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>VarNames</i>	A Variant that specifies the array of variables (memory locations) to be read from the control engine.
<i>VarValues</i>	A Variant that contains an array of the corresponding values of the specified variables in the control engine.
<i>States</i>	A Variant that contains an array of the quality code (Long) for each of the variables.
<i>result</i>	A Long value that indicates whether an error has occurred. The result is zero if no error occurs.

B.55 ReadVariable Method

Applies to: Data

This method reads the status of one specific variable in the control engine.

Syntax:

```
result = object.ReadVariable ( VariableName, Value, State, TimeOut )
```

The ReadVariable method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>VariableName</i>	A String expression that specifies the variable (memory location) in the control engine to be read.
<i>Value</i>	A Variant value containing the content of the specified variable in the control engine.
<i>State</i>	A Long value that provides the quality code for the variable.
<i>TimeOut</i>	A Long value that determines the length of time (in ms) before generating a time-out error. (Not applicable for this release). For the current release, this value should always be 0.
<i>result</i>	A Long value that indicates whether an error has occurred. The result is zero if no error occurs.

B.56 ScaleMode Property

Applies to: Edit, Slider

This property specifies the scaling mode to be used for scaling values. The values can also be specified at the Scaling property tab. There are three choices for scaling mode:

- Scaling by formula (1–wByFormula): $\text{Value} * \text{Factor} + \text{Offset} = \text{DisplayValue}$
where: Value is similar to the contents of the Value property if the control is connected to the control engine; Factor is the value of the Factor property; Offset is the value of the Offset property; and DisplayValue is the contents of the Text property.
- Scaling by range transformation (2–wByRange): you specify a source range (of PLC values) and a destination range (of displayed values), and the values of the one range are transformed to the other range.

Note

The Scale Mode property determines whether the RawMax, RawMin, Factor, and Offset properties have any effect.

Syntax:

object.ScaleMode [= *value*]

The ScaleMode property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the kind of scaling.

The settings for *value* are:

Setting	Description
wNoScaling (0)	(default) No scaling
wByFormula (1)	Use the formula containing the factor and offset to scale the value
wByRange (2)	Use the range transformation method to scale the value

B.57 SelectEvent Method

Applies to: DBuffer

This method selects a specific diagnostic entry in the upper panel of the control.

Syntax:

result = *object.SelectEvent* *EventNumber*

The SelectEvent has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.
<i>EventNumber</i>	An integer that evaluates to the number (event ID) of the diagnostic event.

B.58 ShowErrorBoxes Property

Applies to: Data

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.

Note

Computing provides error messages in English only. If you want to display messages in other languages, you must disable (deselect) the ShowErrorBoxes option and write program code to react on the error event.

Syntax:

object.ShowErrorBoxes [= *value*]

The ShowErrorBoxes property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	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.

B.59 ShowMinMax Property

Applies to: Slider

This property specifies whether the control displays the range (minimum and maximum) of values.

Syntax:

object.ShowMinMax [= *boolean*]

The ShowMinMax property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>boolean</i>	A Boolean expression that specifies whether the control displays the range of values.

The settings for *boolean* are:

Setting	Description
True	(default) The control displays the minimum and maximum vales.
False	The control does not display the range of values.

B.60 SmallChange Property

Applies to: Slider

This property determines how far the indicator moves when the control has focus and you press the up/down or right/left arrow keys. The Value property is increased by SmallChange if you press the right (or up) arrow key. It is decreased by SmallChange if you press the left (or down) arrow key.

Syntax:

object.SmallChange [= *value*]

The SmallChange property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the amount of change

B.61 StretchMode Property

Applies to: Button, Slider, Label

This property returns or sets the stretch mode (centered, resize image, resize frame, smart tile or tile) of the control. This property can only be used if the Style property is set to 1 – wGraphical.

Syntax:

`object.StretchMode [= value]`

The StretchMode property has these parts:

Part	Description
<i>object</i>	The identifier for the specific control
<i>value</i>	A constant that determines the stretch mode, as described in Settings

The settings for *value* are:

Setting	Description
0	wCentered The bitmap is centered in the control.
1	wResizImage (Default) The bitmap is resized (stretched or shrunk) to fit the control.
2	wResizeFrame The frame of the control is resized to the size of the bitmap.
3	wSmartTile The bitmap is expanded to fit the control by replicating adjacent rectangles. This setting works best with a single-color bitmap with a border.
4	wTile The bitmap, if smaller than the control, is duplicated and tiled to fill the control.

B.62 Style Property

Applies to: Button, Slider, Label

This property returns or sets the style (standard or graphical) of the control.

Syntax:

`object.Style [= value]`

The Style property has these parts:

Part	Description
<i>object</i>	The identifier for the specific control
<i>value</i>	A constant that determines the style, as described in Settings

The settings for *value* are:

Setting	Description
0	wStandard (uses internal drawing methods)
1	wGraphical (Default) (uses bitmaps)

B.63 TagSource Property

Applies to: Data

This property identifies the source of symbolic information to be used when assigning variables and identifying control engines. The source can be a tag file. See Appendix G for more information about control engine strings.

Syntax:

`object.TagSource [= value]`

The TagSource property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A String that identifies the pathname to the source (such as a tag file) of symbolic information to be used when configuring the control for variables and control engines.

B.64 Text Property

Applies to: Edit

This property determines the text displayed by the control.

Syntax:

object.Text [= *value*]

The Text property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A String value that specifies the text to be displayed by the control.

B.65 Ticks Property

Applies to: Slider

This property sets the number of ticks, or unit markers, of the control. For example, if Ticks = 10, the scale of the control will be divided into 10 sections.

Syntax:

object.Ticks [= *value*]

The Ticks property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that determines the number of unit markers to be displayed.

B.66 TrueCaption Property

Applies to: Button

This property determines the text that is displayed in the control when the Value property is True (equal to 1, or "On").

Syntax:

object.TrueCaption [= *string*]

The TrueCaption property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>string</i>	Text that determines the active or inactive text of the control

B.67 TrueColor Property

Applies to: Button

This property determines the color of the control when the Value property is True (equal to 1, or "On").

Syntax:

object.TrueColor [= *color*]

The TrueColor property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>color</i>	A value or constant that determines the background or foreground colors of an object, as described in Settings

The settings for *value* are:

Setting	Description
Standard Colors	Colors specified by using the RGB Color palette
Windows System Colors	Colors specified by system color constants (depending on the container); for example, colors listed in the Visual Basic (VB) object library in the Object Browser

B.68 TruePicture Property

Applies to: Button

This property returns or sets the active (on, true, etc.) picture displayed on the control.

Syntax:

object.TruePicture [= *picture*]

The TruePicture property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list
<i>picture</i>	A picture that determines the image of an object

B.69 Update Method

Applies to: DBuffer

This method reads the diagnostic buffer of the control engine and updates the information displayed in the control.

result = *object*.Update

Syntax:

The Update method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>result</i>	A long value that indicates whether an error has occurred. The result is zero if no error occurs.

B.70 Value Property

Applies to: Button, Edit, Slider

This property should be linked, using the DATA Control to a value in the PLC. It is bindable.

Edit Control – The value property is a variant which returns/sets the (unscaled) value of the control.

Button Control – The value property reflects the state of the button.

Slider Control – The value property reflects the position of the Slider Control indicator.

Note

If the value of the Value property changes, the Change event will be generated.

Syntax:

object.Value [= *value*]

The Value property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A Variant that specifies the value of the control.

B.71 WriteMode Property

Applies to: Edit

This property determines how the control responds when the user enters a new value. If the write mode is set to Automatic (0), the value (if valid) is written automatically into the Value property (and to the control engine). If the write mode is Manual (1), the value is not written to the value property unless your program code calls the method "Write" at the control.

Syntax:

object.WriteMode [= *value*]

The WriteMode property has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>value</i>	A value or constant that specifies whether the control automatically passes entered values to the Value property.

The settings for *value* are:

Setting	Description
Automatic (0)	(default) Automatically passes the new (input) value to the Value property
Manual (1)	Does not write the new (input) value unless the control processes a Write method

B.72 WriteNow Method

Applies to: Edit

This method issues a “value changed” for the Value property of the control. You must use this method only if the WriteMode property is set to Manual (1).

Syntax:

object.WriteNow

The WriteNow method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.

B.73 WriteMultiVariables Method

Applies to: Data

This method writes new values for several variables in the control engine.

Syntax:

result = *object*.WriteMultiVariables (*VarNames*, *VarValues*, *States*)

The WriteMultiVariables method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>VarNames</i>	A Variant that specifies the array of variables (memory locations) in the control engine.
<i>VarValues</i>	A Variant that contains an array of the corresponding values to be written to the specified variables.
<i>States</i>	A Variant that contains an array of the quality code (Long) for each of the variables.
<i>result</i>	A Long value that indicates whether an error has occurred. The result is zero if no error occurs.

B.74 WriteVariable Method

Applies to: Data

This method writes a new value to a specific variable in the control engine.

Syntax:

```
result = object.WriteVariable ( VariableName, Value, TimeOut )
```

The WriteVariable method has these parts:

Part	Description
<i>object</i>	An object expression that evaluates to an object in the Applies To list.
<i>VariableName</i>	A String expression that specifies the variable (memory location) in the control engine.
<i>Value</i>	A Variant value containing the content to be written to the specified variable in the control engine.
<i>TimeOut</i>	A Long value that determines the length of time (in ms) before generating a time-out error. (Not applicable for this release). For the current release, this value should always be 0.
<i>result</i>	A Long value that indicates whether an error has occurred. The result is zero if no error occurs.

B.75 Zeropad Property

Applies to: Edit

This property determines whether the number displayed by the control is padded with zeros (to the left of the value) to the size of the data type.

Syntax:

object.ZeroPad [= *value*]

The ZeroPad property has these parts:

Part	Description
<i>object</i>	An object expression (identifier for the specific SIMATIC control) that evaluates to an object in the Applies To list.
<i>value</i>	A Boolean expression that specifies whether or not the displayed number is filled with leading zeros.

The settings for *value* are:

Part	Description
True	Fills the number with leading zeros to the size specified by the DataType property.
False	(default) Does not fill the number with leading zeros.

Events

C.1 Change Event

Applies to: Button, Edit, Label, Slider

This event occurs when the value of the Value property changes. Either the control engine or the S7 control object can change the value in the Value property.

Syntax: **Change** ()

C.2 Click Event

Applies to: Button, Edit, Label, DBuffer, Slider

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

Syntax: **Click** ()

Note

To distinguish between the left, right, and middle mouse buttons, use the MouseDown and MouseUp events. If there is code in the Click event, the DbClick event will never trigger, because the Click event is the first event to trigger between the two. As a result, the mouse click is intercepted by the Click event, so the DbClick event does not occur.

C.3 ConnectionError Event

Applies to: Data

This event occurs when an error on a connection occurs.

Syntax:

```
ConnectionError(State As Long, ConnectedObject As Object, _  
Property As String, Variable As String)
```

The ConnectionError event has these parts:

Part	Description
<i>State</i>	A long value with the state of the connection
<i>ConnectedObject</i>	An object expression that evaluates to the connected object
<i>Property</i>	A string value with the name of the property
<i>Variable</i>	A string value with the name of the connected variable

C.4 DbClick Event

Applies to: Edit, Label, Slider

This event occurs when a mouse button is double-clicked while the cursor is over the control.

Syntax: **DbClick()**

Note

To distinguish between the left, right, and middle mouse buttons, use the MouseDown and MouseUp events.

If there is code in the Click event, the DbClick event will never trigger, because the Click event is the first event to trigger between the two. As a result, the mouse click is intercepted by the Click event, so the DbClick event does not occur.

C.5 Error Event

Applies to: Button, Edit, Label, Slider

This event occurs when the control encounters an error.

Syntax:

Error(long SCode, BSTR lpszDescription, BSTR lpszHelpFileName, _
long nHelpId)

The Error event has these parts:

Part	Description
<i>SCode</i>	See Table C-1
<i>lpszDescription</i>	String with a description of the error condition

Part	Description
<i>lpSzHelpFileName</i>	Name of the Help file in which the error is described
<i>nHelpId</i>	Help topic ID with a description of the error

Table C-1 SCodes (Error Event Codes)

Name	Value	Description
wFACTOR_ZERO	0xC0040002	Factor: Must not be zero.
wRAWMINMAX	0xC0040006	RawMin must be less than RawMax.
wMINMAX	0xC0040009	Min must be less than Max.
wLARGECHANGE_ZERO	0xC004000A	Large Change: Must be greater than zero and less than...
wTICKS_ZERO_100	0xC004000C	Ticks: Must be a number between 1 and 100.
wKNOBHEIGHT_ZERO	0xC004000E	Knob Height: Must be greater than zero.
wKNOBWIDTH_ZERO	0xC0040010	Knob Width: Must be greater than zero.
wSMALLCHANGE_ZERO	0xC0040012	Small Change: Must be greater than zero and less than...
wRAWMIN_SCALEMODE	0xC0040014	RawMin may only be set if ScaleMode is wByRange.
wRAWMAX_SCALEMODE	0xC0040015	RawMax may only be set if ScaleMode is wByRange.
wEDIT_OUT_OF_RANGE	0xC0040016	Value out of range.
wEDIT_WRONGVALUE	0xC0040017	A wrong value has been set.
wBIGFONT	0xC0040018	Warning: Font size is too big.
wPREC_RANGE	0xC004001A	Precision: Must be a number between 0 and 7.

C.6 KeyDown Event

Applies to: Button, Edit, Slider

This event occurs when the user presses a key while the control has the focus. See also the KeyUp Event.

Syntax: **KeyDown**(*long KeyID*, *long Shift*)

The KeyDown event has these parts:

Part	Description
<i>KeyID</i>	Key code, such as <code>vbKeyF1</code> (the F1 key) or <code>vbKeyHome</code> (the HOME key) To specify key codes, use the constants in the Visual Basic (VB) object library in the Object Browser.
<i>Shift</i>	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event 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. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys have been pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use the `KeyDown` and `KeyUp` event procedures if you need to respond to both the pressing and releasing of a key.

`KeyDown` and `KeyUp` interpret the uppercase and lowercase of each character by means of two arguments: `keycode`, which indicates the physical key (thus returning A and a as the same key) and `shift`, which indicates the state of shift+key and therefore returns either "A" or "a".

If you need to test for the shift argument, you can use the shift constants that define the bits within the argument. The constants have the following values:

- `vbShiftMask` (1): SHIFT key bit mask
- `vbCtrlMask` (2): CTRL key bit mask
- `vbAltMask` (4): ALT key bit mask

The constants act as bit masks that you can use to test for any combination of keys.

You test for a condition by first assigning each result to a temporary integer variable and then comparing `Shift` to a bit mask. Use the `And` operator with the `Shift` argument to test whether the condition is greater than 0, indicating that the modifier was pressed.

C.7 KeyPress Event

Applies to: Button, Edit, Slider

This event occurs when an ANSI key is pressed and released while the control has the focus.

Syntax: `KeyPress (long keyAscii)`

The `KeyPress` event has these parts:

Part	Description
<i>keyAscii</i>	ASCII key code of the pressed key, such as vbKeyF1 (the F1 key) or vbKeyHome (the HOME key)

C.8 KeyUp Event

Applies to: Button, Edit, Slider

This event occurs when a key is released while the control has the focus.

Syntax: `KeyUp(long KeyID, long Shift)`

The KeyDown event has these parts:

Part	Description
<i>KeyID</i>	Key code, such as vbKeyF1 (the F1 key) or vbKeyHome (the HOME key) To specify key codes, use the constants in the Visual Basic (VB) object library in the Object Browser.
<i>Shift</i>	An integer that corresponds to the state of the SHIFT, CTRL, and ALT keys at the time of the event 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. Some, all, or none of the bits can be set, indicating that some, all, or none of the keys have been pressed. For example, if both CTRL and ALT are pressed, the value of shift is 6.

Use KeyDown and KeyUp event procedures if you need to respond to both the pressing and releasing of a key.

KeyDown andKeyUp interpret the uppercase and lowercase of each character by means of two arguments: keycode, which indicates the physical key (thus returning A and a as the same key) and shift, which indicates the state of shift+key and therefore returns either "A" or "a".

If you need to test for the shift argument, you can use the shift constants which define the bits within the argument. The constants have the following values:

- vbShiftMask (1): SHIFT key bit mask
- vbCtrlMask (2): CTRL key bit mask
- vbAltMask (4): ALT key bit mask

The constants act as bit masks that you can use to test for any combination of keys.

You test for a condition by first assigning each result to a temporary integer variable and then comparing Shift to a bit mask. Use the And operator with the Shift argument to test whether the condition is greater than 0, indicating that the modifier was pressed.

C.9 MouseDown Event

Applies to: Button, Edit, Label, Slider

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
<i>Button</i>	<p>An integer that identifies the button that was pressed to cause the event</p> <p>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.</p>
<i>Shift</i>	<p>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</p> <p>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.</p>
<i>x,y</i>	returns a number that specifies the current location of the mouse pointer

C.10 MouseMove Event

Applies to: Button, Edit, Label, Slider

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
<i>Button</i>	<p>An integer that identifies the button that was pressed to cause the event</p> <p>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.</p>
<i>Shift</i>	<p>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</p> <p>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.</p>
<i>x,y</i>	<p>returns a number that specifies the current location of the mouse pointer</p>

C.11 MouseUp Event

Applies to: Button, Edit, Label, Slider

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
<i>Button</i>	<p>An integer that identifies the button that was pressed to cause the event</p> <p>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.</p>
<i>Shift</i>	<p>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</p> <p>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.</p>
<i>x,y</i>	returns a number that specifies the current location of the mouse pointer

C.12 ValueChanged Event

Applies to: Data

This event occurs when the value of a connected variable changes and no connected event was specified on the call to the Connect method. A ValueChangedEvent can also be configured using the Events Property Tab.

Syntax:

```
ValueChanged(Property As String, Variable As String, Value as _
Variant, Quality as Integer)
```

The ValueChanged event has these parts:

Part	Description
<i>Property</i>	A string value with the name of the property
<i>Variable</i>	A string value with the name of the connected variable
<i>Value</i>	A variant with the new value of Variable
<i>Quality</i>	Returns an integer with the quality of the new value

Using the Computing Configuration Tool

D

Chapter Overview

The Computing Configuration tool allows you to direct communications to a control engine. You can also use the tool to choose a language for the Computing Software, to set up the OPC server, or to connect applications generated from older versions of Computing that did not support tag files.

Note

You can have only one control engine active at a time. Instead of reconfiguring the Data Control in your program, you can use the configuration tool to change the control engine for the connection.

Section	Description	Page
D.1	Configuring the OPC Connection	D-2
D.2	Selecting the Language	D-5
D.3	Selecting the Control Engine for Older Programs	D-6
D.4	Setting Up Communications Using the PG/PC Interface	D-7

D.1 Configuring the OPC Connection

Computing allows you to use OPC for connecting either to a single control engine or to several control engines. You can also connect to the control engine over a network, such as a local area network (LAN).

As shown in Figure D-1, the OPC Setup application provides the following options for connecting to a control engine:

- The “Connection via Tag Source” option allows you to connect simultaneously to several control engines. You enter (or browse to) the tag file that identifies symbols for the variables and control engines to be accessed.
- The “Direct Connection” option allows you to connect to a specific control engine on a specific computer. With a direct connection, you cannot connect to a tag file and use symbols to access data in the control engine. For this option, you specify the name of the specific control engine and the name of the target computer.

Use the following strings in the “Control Engine” field to define different SIMATIC PLCs:

- **winLC** (for the WinLC of WinAC Basis)
- **wcS7=3** (for a slot PLC such as the CPU 416-2 DP ISA of WinAC Pro)
- **wcS7=xx,a,b** (for other SIMATIC PLCs on the MPI network, where *xx* is the MPI address, *a* is the rack number, and *b* is the slot number)
- **wcIP=xxx.xxx.xxx.xxx,a,b** for a control engine on a TCP/IP network or **wcMAC=xx.xx.xx.xx.xx.xx,a,b** for a control engine on an Industrial Ethernet with STEP 7 v3 SP3

Refer to Appendix G for more information about control strings.

Configuring the OPC Connection

Use the following procedure to configure the OPC connections with the WinAC OPC Setup application:

1. Select the **Simatic ► PC Based Control ► Computing Configuration** menu command from the Start menu to display the “Computing Configuration” dialog box. Select the OPC tab. See Figure D-1.
2. To connect to a specific control engine without using symbols, select the “Direct Connection” option and enter the name of the target computer and the name of the control engine.

3. To use symbols for accessing data in the control engine or to access multiple control engines, select the “Connection via Tag Source” option and enter the name of the tag file. (Click on the “Browse” button to browse to the tag file.)
4. Click on the “Apply” button to enter the data, and click on the “OK” button to close the dialog box.

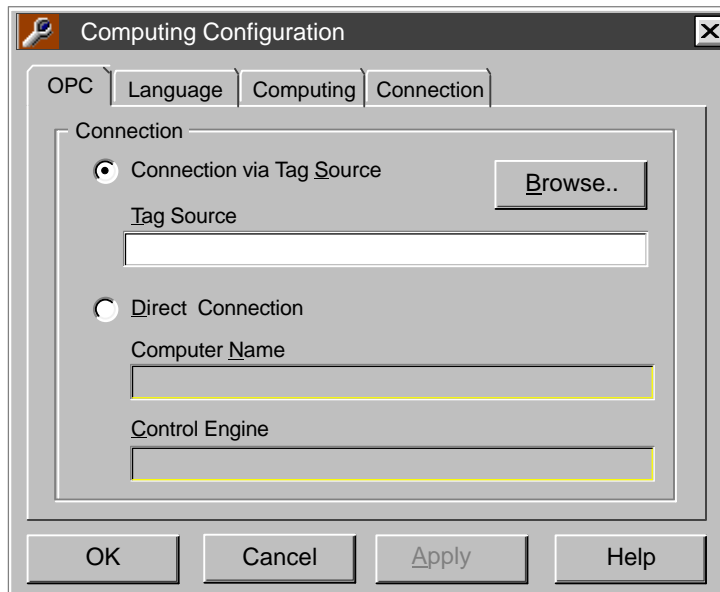


Figure D-1 Configuring the OPC Connection

OPC Error Codes

Table D-1 lists the error codes for the OPC interface. OPC methods return error codes in a HRESULT (a long variables, in hexadecimal format). For Visual C, error conditions are handled with the HRESULT. For Visual Basic, error handling is written to the VB error object (ERR). You must add code to your VB program to access the error codes from the OPC interface.

Table D-1 OPC Error Codes

Error Code	Error	Description
0x80070057	E_INVALIDARG	The value of one or more parameters was not valid. This is generally used in place of a more specific error where it is expected that problems are unlikely or will be easy to identify (for example, when there is only one parameter).
0x8007000E	E_OUTOFMEMORY	There is not enough memory to complete the requested operation. This can happen any time the server needs to allocate memory to complete the requested operation.
0x0004000D	OPC_E_UNSUPPORTEDRATE	The server does not support the requested data rate, but will use the closest available rate.
0x0004000E	OPC_E_CLAMP	A value passed to WRITE was accepted, but was clamped.
0xC0040001	OPC_E_INVALIDHANDLE	An invalid handle was passed.
0xC0040002	OPC_E_DUPLICATE	A duplicate parameter was passed where one is not allowed.
0xC0040003	OPC_E_UNKNOWNLCID	The server does not support the specified local ID.
0xC0040004	OPC_E_BADTYPE	The server cannot convert between the passed or requested data type and the canonical data type for this item.
0xC0040005	OPC_E_PUBLIC	The requested operation cannot be performed on a public group.
0xC0040006	OPC_E_BADRIGHTS	The access rights for the item do not allow the operation.
0xC0040007	OPC_E_UNKNOWNITEMID	The item definition does not exist within the address space of the server. This can also occur on an exiting item if the item is deleted "on-line" from the server address space by some external operation.
0xC0040008	OPC_E_INVALIDITEMID	The item definition does not conform to the syntax of the server.
0xC0040009	OPC_E_INVALIDFILTER	The filter string is not valid.
0xC004000A	OPC_E_UNKNOWNPATH	The access path of the item is not known to the server.
0xC004000B	OPC_E_RANGE	A value passed to WRITE is out of range.
0xC004000C	OPC_E_DUPLICATE_NAME	A group with a duplicate name already exists in the server.

D.2 Selecting the Language

SIMATIC Computing provides three languages for the software and help: German, English, and French. The menus and help are displayed in the language selected. Use the following procedure to change the language for SIMATIC Computing:

1. Select the **Simatic ► PC Based Control ► Computing Configuration** menu command from the Start menu to display the “Computing Configuration” dialog box.
2. In the “Computing Configuration” dialog box, select the “Language” tab.
3. Select the language for the CPU panel (German, English, or French). See Figure D-2.
4. Click on the “Apply” button to change the language.
5. Click on the “OK” button to close the “Customize” dialog box.

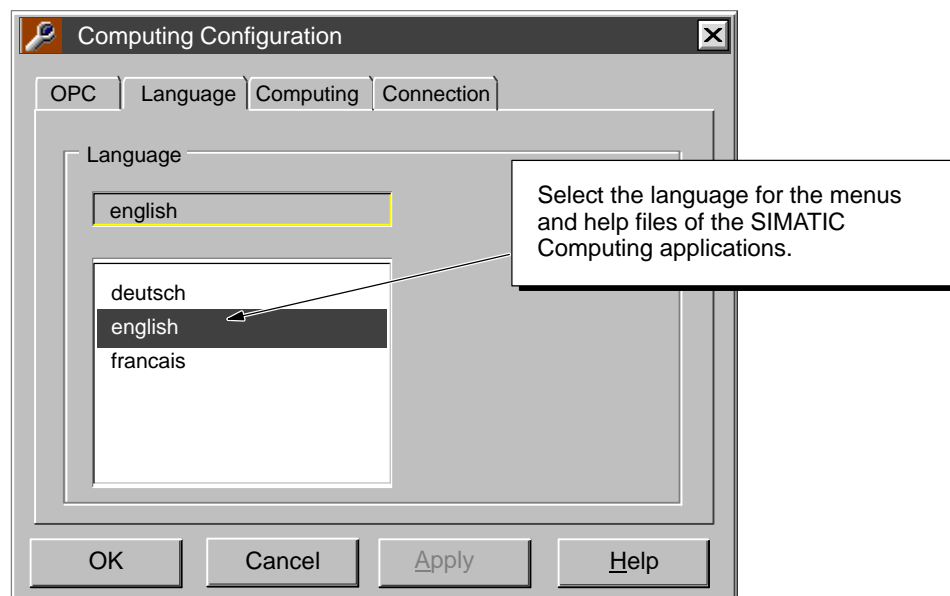


Figure D-2 Selecting the Language for the CPU Panel and Help Files

D.3 Selecting the Control Engine for Older Programs

The “Computing” tab of the configuration tool provides compatibility with programs which were created with earlier versions of Computing that used WinAC\Default to access the local control engine. (WinAC\Default does not work with remote computers or with multiple control engines.) For older programs, use the following procedure to select the local control engine:

Note

This page only provides backward compatibility with applications generated from older Computing versions that did not support tagfiles. If you are using tagfiles, you do not need to use this page.

1. Use the Start menu (**Start ► Simatic ► PC Based Control ► Computing Configuration**) to open the Computing Configuration tool.
2. Click on the “Computing” tab.
3. As shown in Figure D-3, select the control engine:
 - Click on the “WinLC” option to select the WinLC control engine.
 - Click on the “CPU416-2 DP ISA” option to select the CPU 416-2 DP ISA control engine.
 - Click on the “MPI” option and enter an MPI address to select a PLC on the MPI network as the control engine.
4. Click on the “OK” button to select the control engine. (Click on the “Undo” button to reset the control engine.)

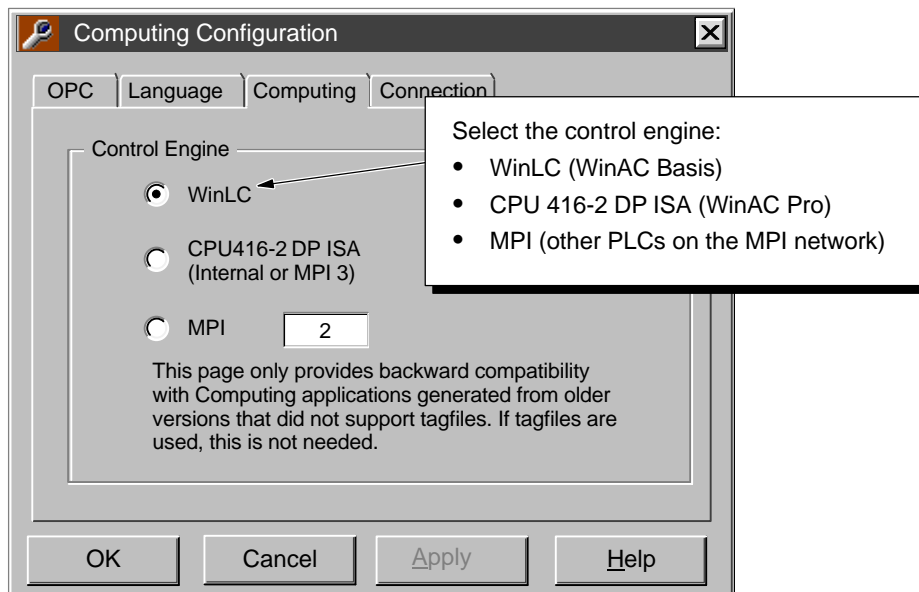


Figure D-3 Selecting the Control Engine

D.4 Setting up Communications Using the PG/PC Interface

This tab provides access to Setting the PG/PC Interface, which you use to set up communications with WinLC and other PLCs across MPI, PROFIBUS-DP, and H1 networks.

1. Use the Start menu (**Start ► Simatic ► PC Based Control ► Computing Configuration**) to open the Computing Configuration tool.
2. Click on the “Connection” tab.

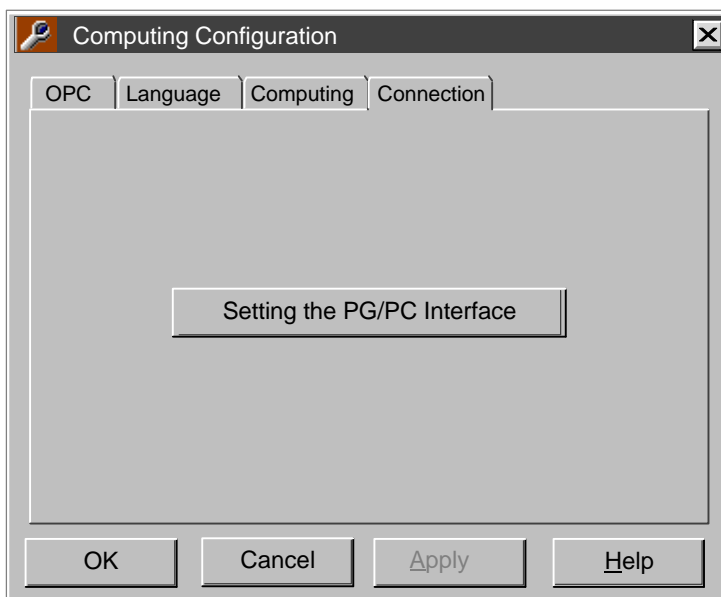


Figure D-4 Accessing the PG/PC Interface

Connecting STEP 7 to WinLC on the Same Computer

1. To configure STEP 7 as the local access point, follow the steps below:
2. From the “Access point of application” drop-down list, select **S7ONLINE (STEP 7)** (Figure D-5).
3. From the “Interface parameter set used” drop-down list, select **PC Internal (local)** for the interface parameter.

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

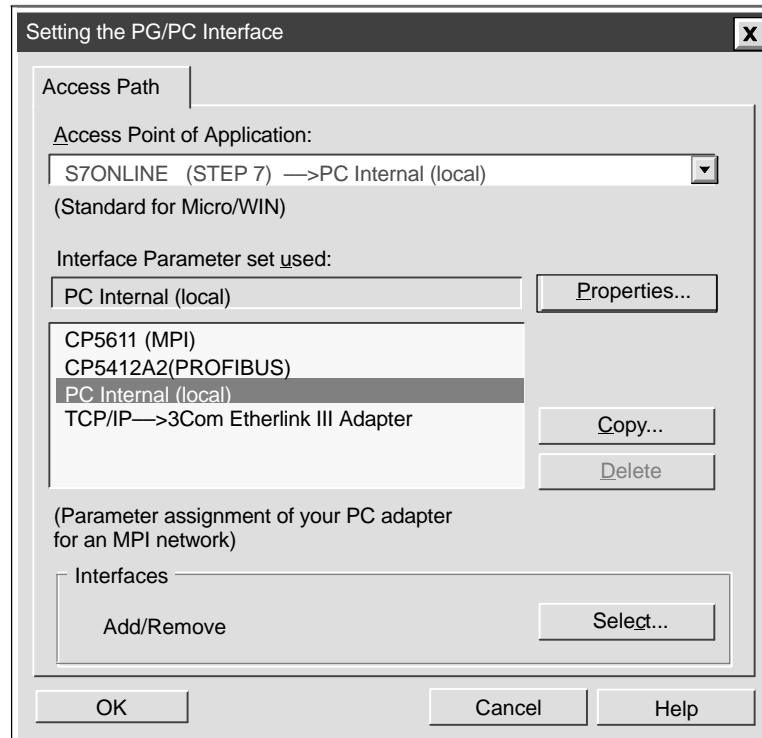


Figure D-5 Setting the PG/PC Interface for PC Internal (local)

Connecting STEP 7 to WinLC on a Different Computer

To connect STEP 7 on one computer to a WinLC on a different computer, you must define the network connection over which STEP 7 and WinLC communicate by setting the PG/PC interface on the remote computer.

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

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

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

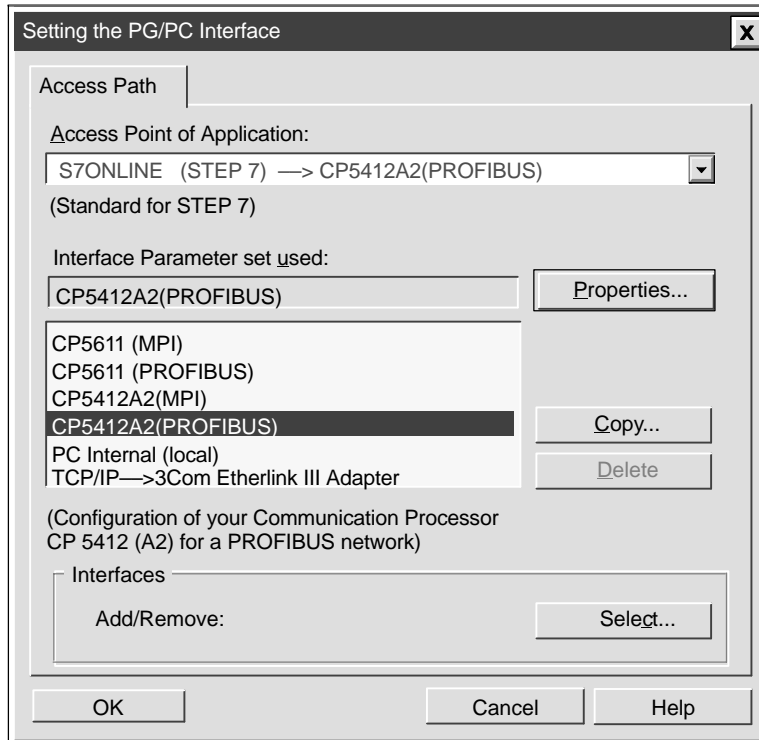


Figure D-6 Setting the PG/PC Interface From the STEP 7 Computer

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

The WinLC’s PROFIBUS card must be properly configured through **Setting the PG/PC Interface** before WinLC is visible to other PGs on the PROFIBUS–DP network (**S7ONLINE (STEP7) -> Profibus....** It must be set for “PG is the only master on the bus.”

- For Industrial Ethernet communication, select the TCP/IP interface, for example, **TCP/IP -> 3Com Etherlink III Ada...** You must have the NCM Options package for H1 communication and STEP 7 V5 SP3.

Note

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

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

Example:

```
WinLC_0 —> none
WinLC_1 —> CP5412A2 (PROFIBUS)
WinLC_2 —> none
WinLC_3 —> none
WinLC_4 —> none
WinLC_5 —> none
WinLC_6 —> CP5611 (PROFIBUS)
WinLC_7 —> none
WinLC_8 —> none
WinLC_9 —> none
```

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

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

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

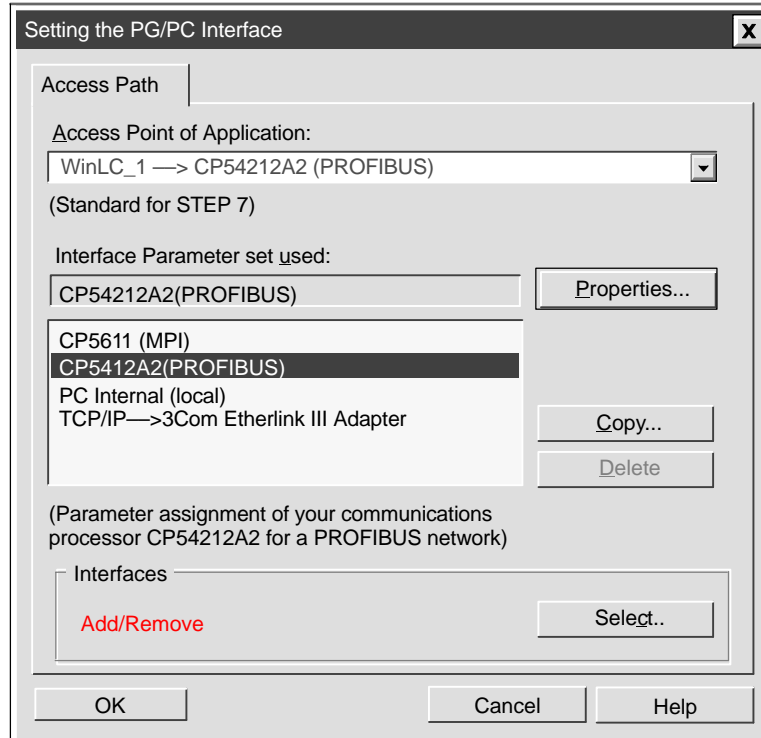


Figure D-7 Setting the PG/PC Interface From WinLC

2. From the “Access Point of Application” drop-down list, select **WinLC_0**.
3. Select the interface parameter from the parameter set that corresponds to your network communications path, for example **CP5412A2 (PROFIBUS)**.

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

Connecting STEP 7 to Hardware PLCs

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

Note

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

Using Computing with DCOM

Chapter Overview

The Computing software allows you to communicate across networks using Windows NT's Distributed Component Object Model (DCOM). You can use DCOM to integrate distributed applications by way of a network. A distributed application consists of multiple processes or different computers that cooperate to accomplish a single task.

DCOM is a set of Microsoft concepts and program interfaces in which client program objects can request services from server program objects on other computers in a network. The Component Object Model (COM) provides a set of interfaces that allow clients and servers to communicate within the same computer (running Windows 95 or Windows NT).

Note

The control engine must be installed on the server computer. If you plan to use the SIMATIC controls provided with Computing to access the control engine, install the Computing software on both the server computer and the client computer.

Section	Description	Page
E.1	Using DCOM to Provide Remote Access	E-2
E.2	Configuring the Permissions for the Server Computer	E-4
E.3	Configuring the Permissions for the Client Computer	E-14
E.4	Troubleshooting	E-20

E.1 Using DCOM to Provide Remote Access

DCOM is a set of Microsoft concepts and program interfaces in which client program objects can request services from server program objects on other computers in a network. The Component Object Model (COM) provides a set of interfaces that allow clients and servers to communicate within the same computer (running Windows 95 or Windows NT).

As described in Section 4.1, you can run Computing on a stand-alone computer, as shown in Figure E-1. In this model, this computer provides the complete control system. Computing allows you to access not only the WinLC of WinAC Basis and a slot PLC such as the CPU 416-2 DP ISA of WinAC Pro, but also other PLCs on a network.

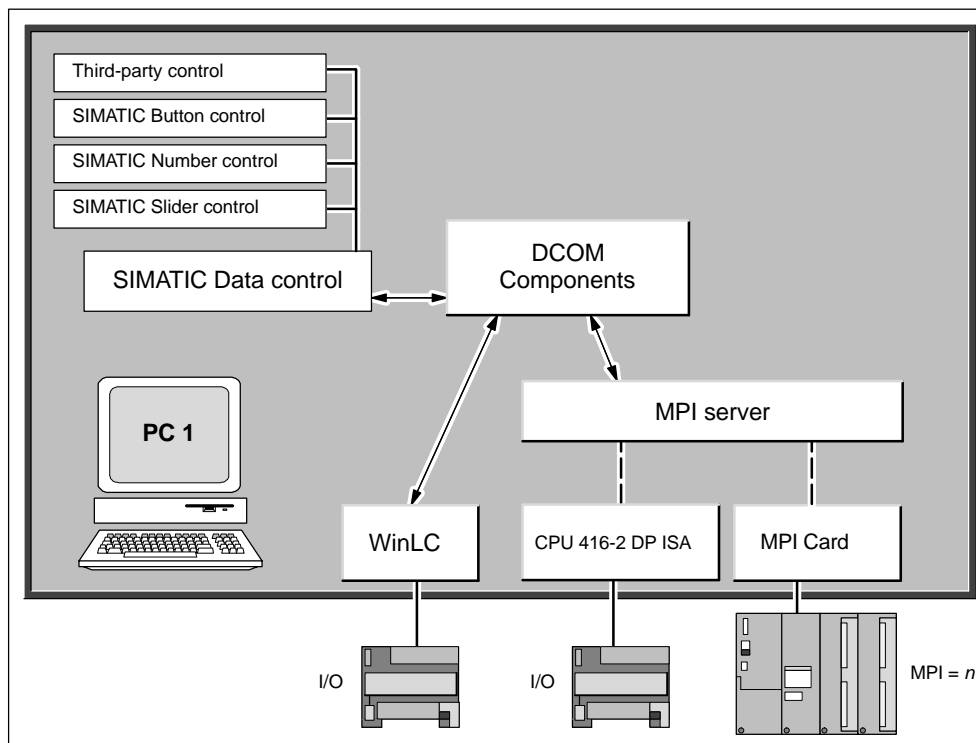


Figure E-1 Using SIMATIC Computing on a Single Computer

You can also utilize Microsoft's DCOM technology to create a network of computers that cooperate to provide the control system for a machine or process. Figure E-2 shows how one computer running an application that uses ActiveX controls (from Computing) can use DCOM to communicate with a different computer that uses WinLC (or other S7 PLCs) to control a process.

The Windows NT operating system provides a configuration tool (dcomcnfg) for setting up your DCOM network. Use this tool to configure the server and client computers. For information about configuring the server computer, see Section E.2; for information about configuring the client computer, see Section E.3.

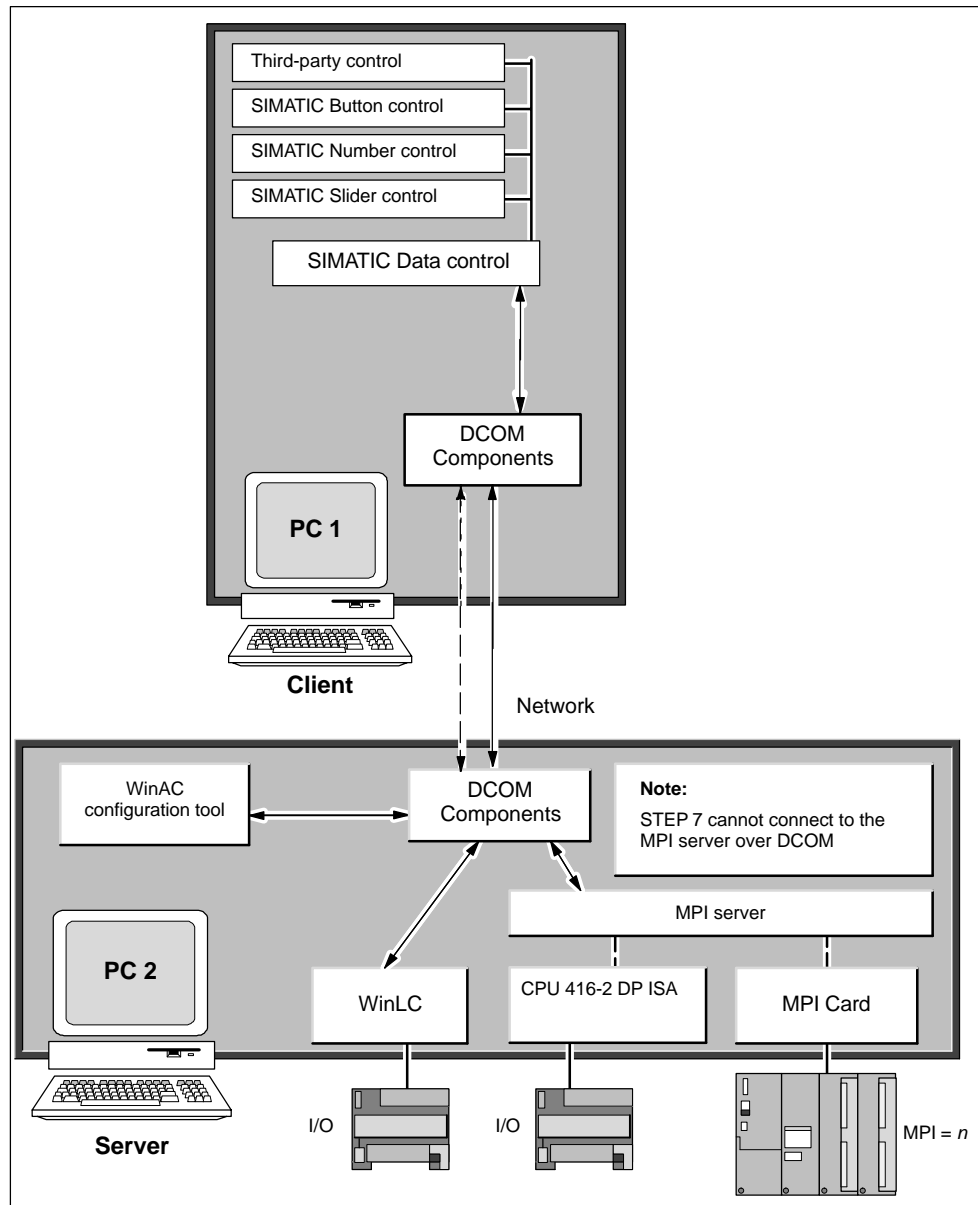


Figure E-2 Using SIMATIC Computing over DCOM

Note

You install the WinAC authorization on the server computer; you install the SIMATIC Computing authorization on the client computer. If you want to run SIMATIC Computing on a PC other than the PC running WinLC, then you must purchase SIMATIC Computing standalone. For more information about installing an authorization, see Section 3.2.

E.2 Configuring the Permissions for the Server Computer

The DCOM network consists of a server computer (where the control engine resides) and one or more client computers. Windows NT provides a configuration tool for setting up the network parameters, such as security and access privileges. For the server application, you must specify the user account that will have permission to access or start the application, and the user accounts that will be used to run the application. This protects your process from unauthorized access. Figure E-3 lists the basic tasks required for configuring the server.



Caution

Granting permission to access applications on a computer allows other users to start and stop programs or to access files on your computer. Granting unlimited access to everyone on the network could cause problems from either innocent or malicious interference. Always limit access to those users who are required to use the applications or files on the computer.

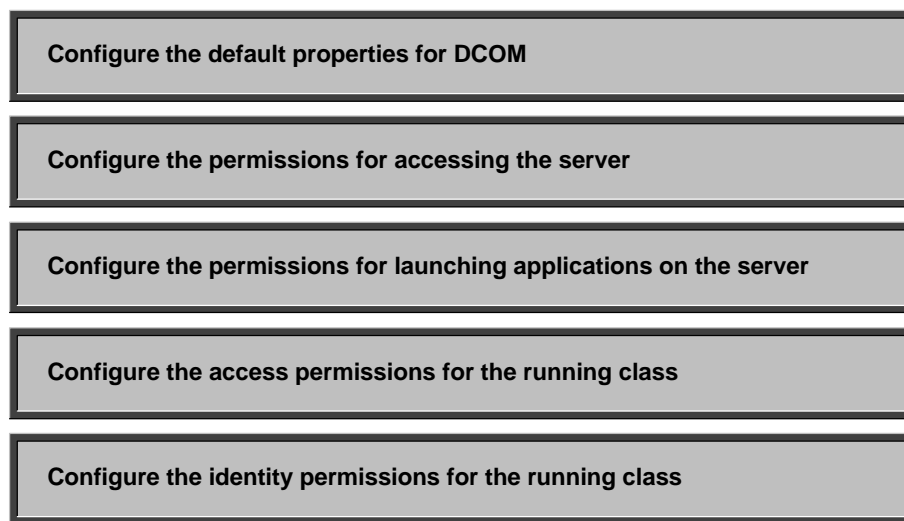


Figure E-3 Tasks for Configuring the DCOM Server

Starting the DCOM Configuration Editor

To configure the DCOM server, you must run the DCOM configuration tool on the computer that will function as the server. Use the following procedure to start the DCOM configuration tool:

1. Select the **Start ► Run...** menu command from the Windows Start menu.
2. In the "Run" dialog box, enter `dcomcnfg` and click on the "OK" button.

The DCOM configuration tool displays the "Distributed COM Configuration Properties" dialog box.

Configuring the Default Properties for DCOM Communication

Use the “Distributed COM Configuration Properties” dialog box to configure the properties of the computer for DCOM. See Figure E-4.

1. Click on the “Default Properties” tab.
2. Select the “Enable Distributed COM on this computer” option.
3. Set the “Default Authentication Level” to the “Connect” option.
4. Set the “Default Impersonation Level” to the “Identify” option.

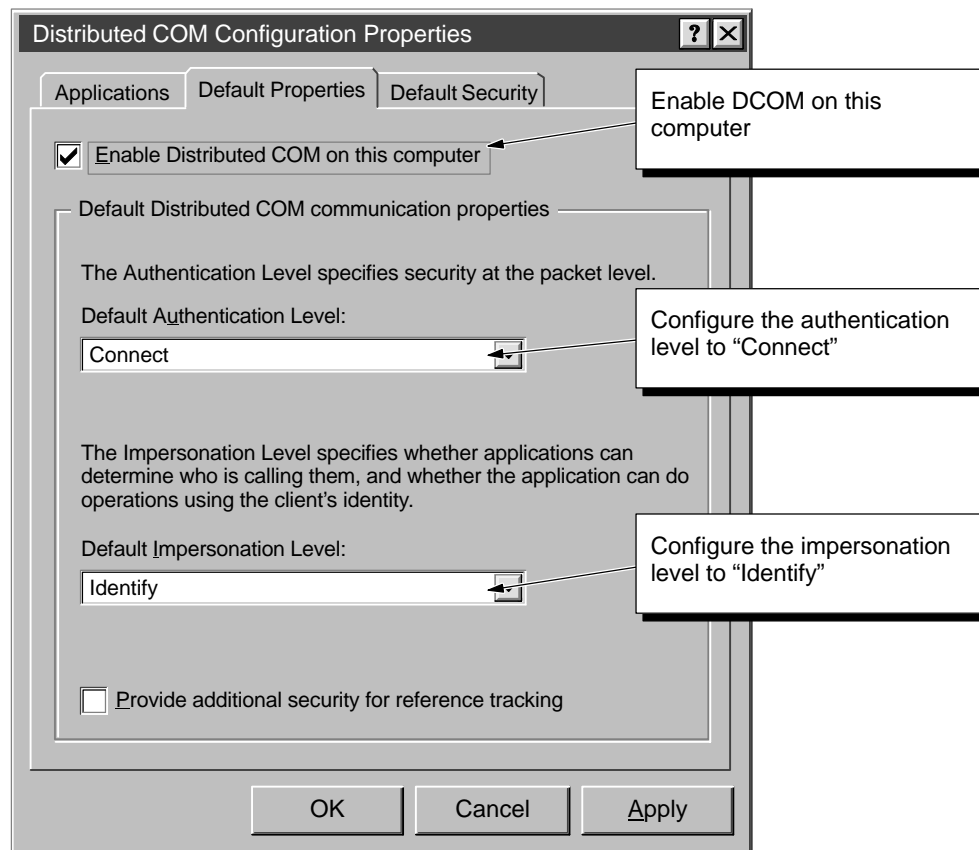


Figure E-4 Distributed COM Configuration Properties

Configuring the Permissions for Accessing Software on the Server

1. Click on the “Default Security” tab to display the security options for DCOM. See Figure E-5.
2. Click on the “Edit Default” button for “Default Access Permissions” to display the “Registry Value Permissions” dialog box.

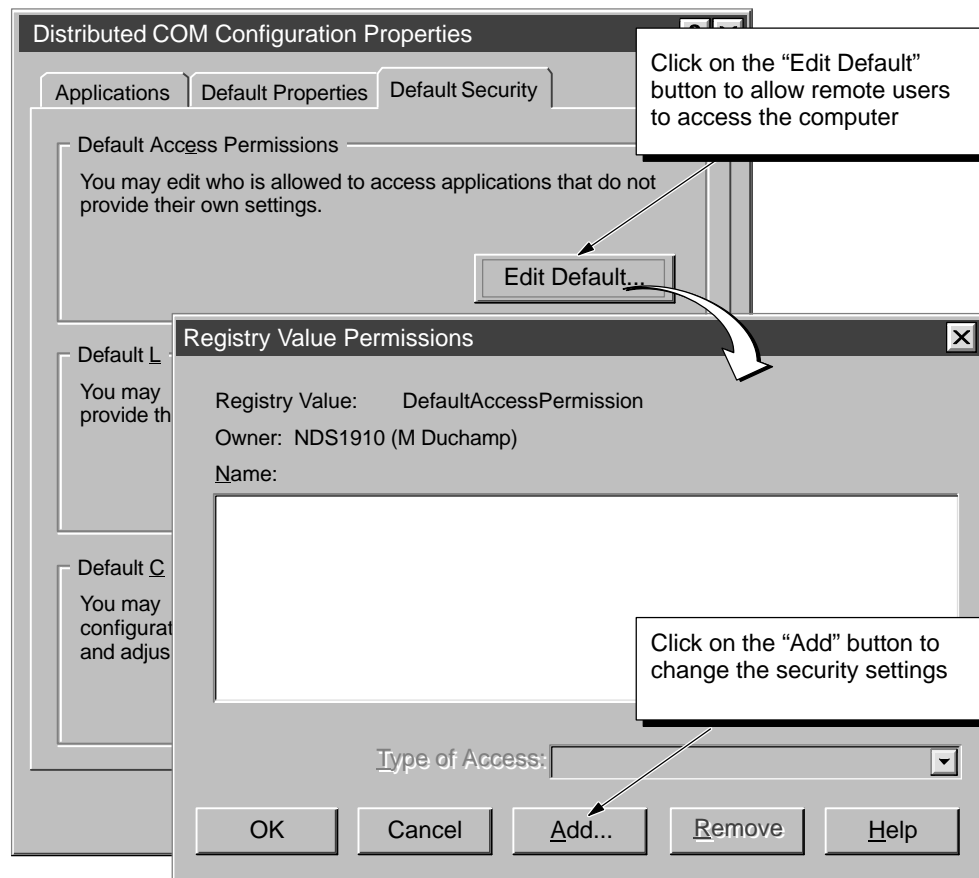


Figure E-5 Configuring the Default Access Permissions for DCOM

3. Click on the “Add” button to display the “Add Users and Groups” dialog box and change the security settings for access to the server. See Figure E-6.
4. From the “Names” field, select “Everyone” (or the appropriate subset of users) and click on the “Add” button.
5. Select “INTERACTIVE” and click on the “Add” button.
6. Select “SYSTEM” and click on the “Add” button.

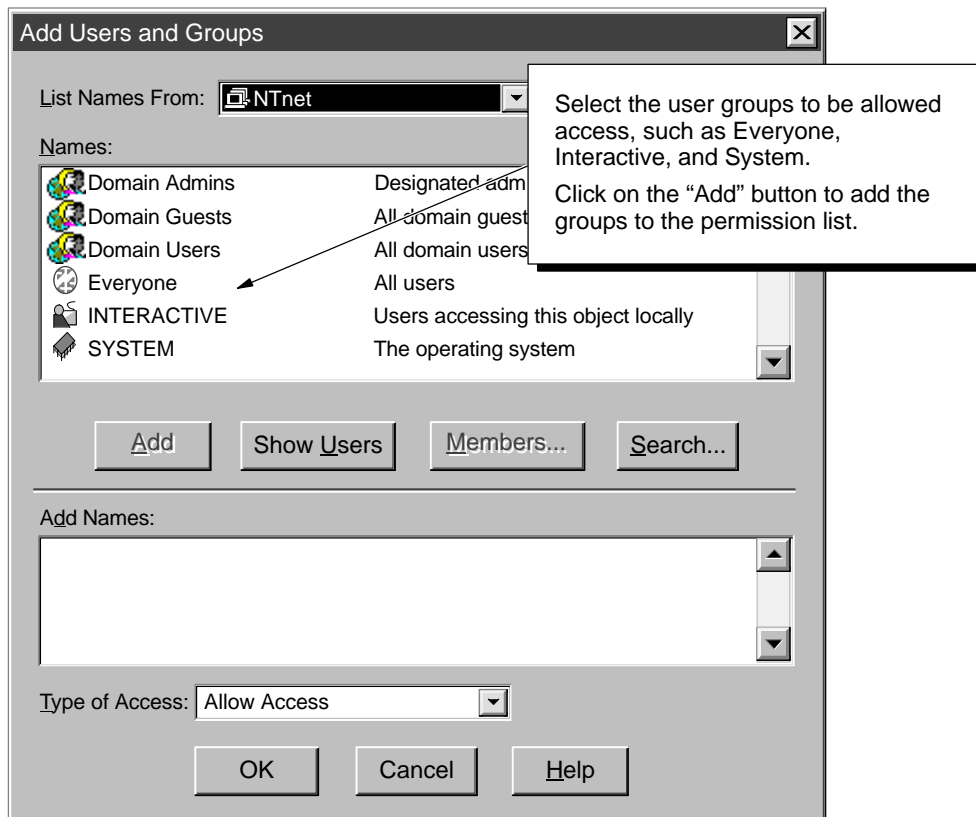


Figure E-6 Changing the Access Permissions for Users or Groups

7. Click on the “OK” button to enter these changes to the “Registry Value Permissions” dialog box.
8. Click on the “OK” button of the “Registry Value permissions” dialog box to enter the changes to the default access permissions. The “Registry Value permissions” dialog box closes and displays the “Distributed COM Configuration Properties” dialog box (Figure E-5).

**Caution**

Granting permission to access applications on a computer allows other users (such as “Everyone”) to start and stop programs or to access files on your computer. Granting unlimited access to everyone on the network could cause problems from either innocent or malicious interference. Always limit access to those users who are required to use the applications or files on the computer.

Configuring the Permissions for Launching Software on the Server

1. Click on the “Edit Default” button for “Default Launch Permissions” to display the “Registry Value Permissions” dialog box. See Figure E-7.
2. Click on the “Add” button to display the “Add Users and Groups” dialog box and change the security settings for access to the server. See Figure E-8.

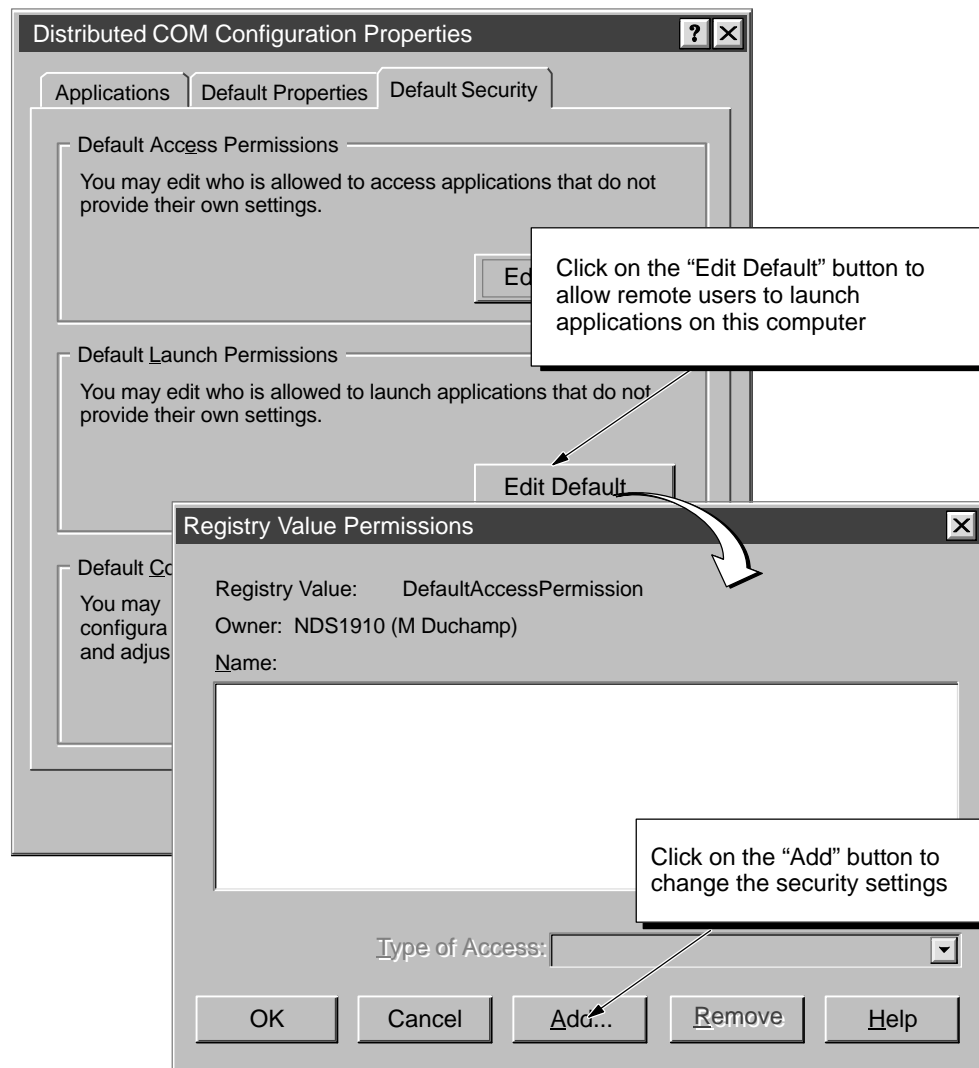


Figure E-7 Configuring the Default Launch Permissions for DCOM

3. In the “Names” field of the “Add users and Groups” dialog box (Figure E-8), select “Everyone” (or the appropriate subset of users) and click on the “Add” button.



Caution

Granting permission to access applications on a computer allows other users (such as “Everyone”) to start and stop programs or to access files on your computer. Granting unlimited access to everyone on the network could cause problems from either innocent or malicious interference. Always limit access to those users who are required to use the applications or files on the computer.

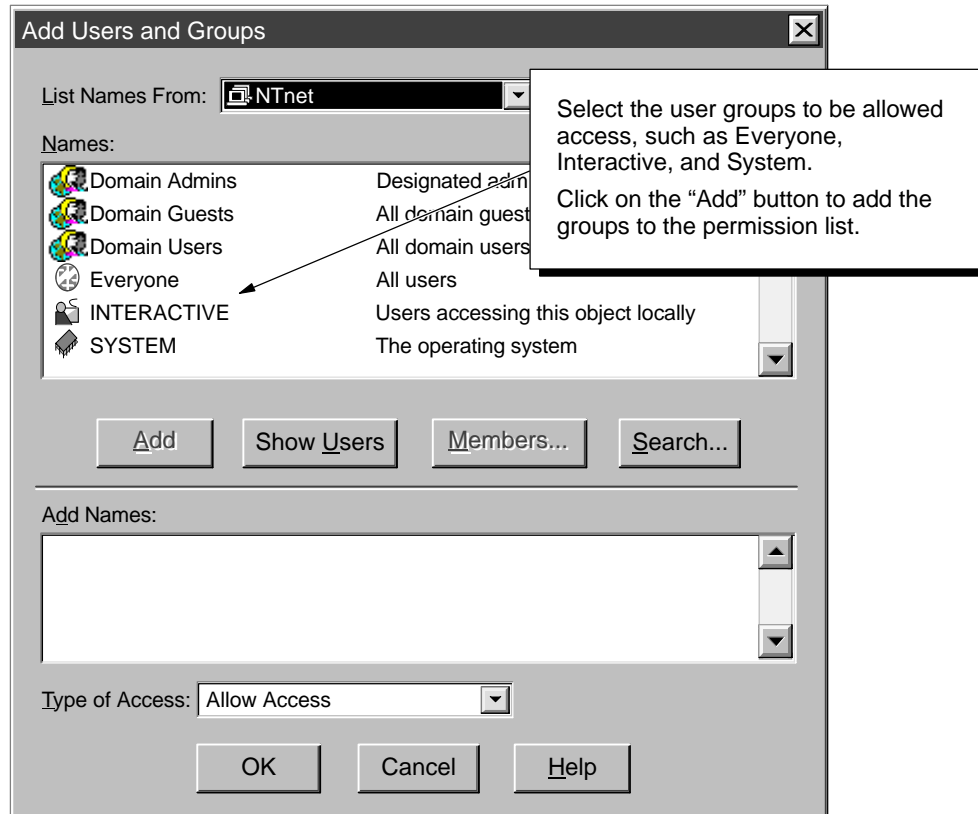


Figure E-8 Changing the Launch Permissions for Users or Groups

4. Select “INTERACTIVE” and click on the “Add” button.
5. Select “SYSTEM” and click on the “Add” button.
6. Click on the “OK” button to enter these changes to the “Registry Value Permissions” dialog box.
7. Click on the “OK” button of the “Registry Value permissions” dialog box to enter the changes to the default access permissions. The “Registry Value permissions” dialog box closes and displays the “Distributed COM Configuration Properties” dialog box.

Configuring the Properties for the Running Class

Use the following procedure to configure the properties of the running class for the server:

1. Click on the “Applications” tab of the “Distributed COM Configuration Properties” dialog box. See Figure E-9.
2. Select “Running Class” from the list of applications.
3. Click on the “Properties” button to display the “Running Class Properties” dialog box.

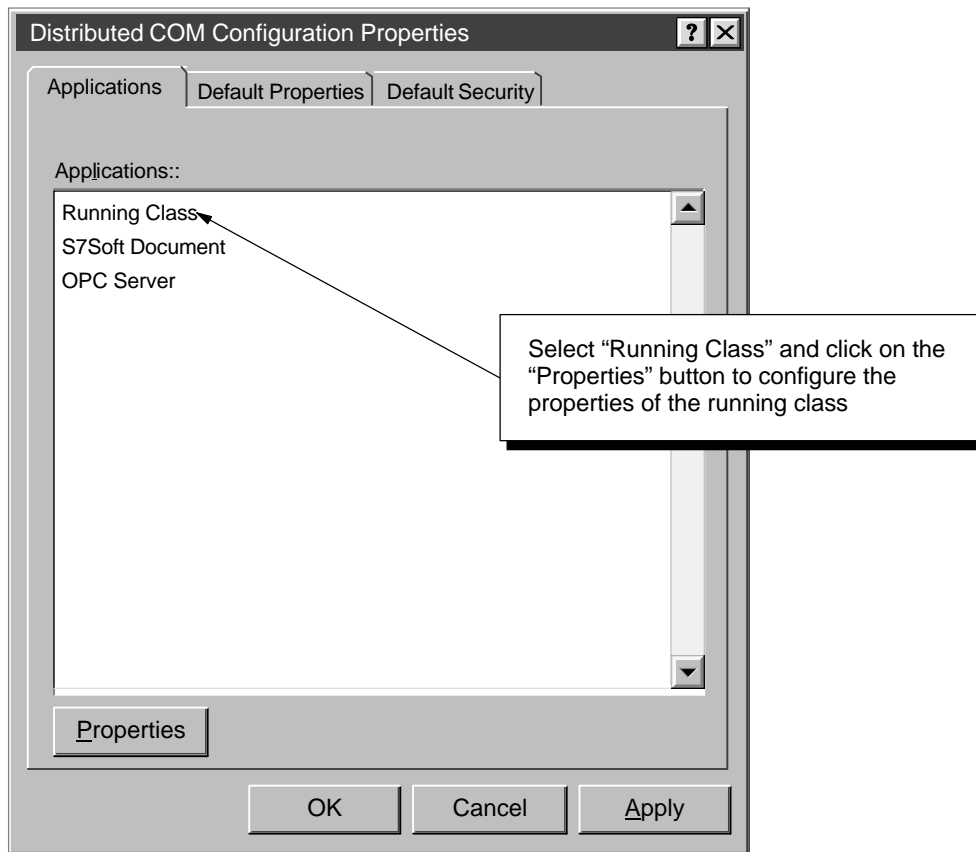


Figure E-9 Selecting the Running Class for DCOM

Configuring the Access Permissions for the Running Class



Caution

Granting permission to access applications on a computer allows other users (such as “Everyone”) to start and stop programs or to access files on your computer. Granting unlimited access to everyone on the network could cause problems from either innocent or malicious interference. Always limit access to those users who are required to use the applications or files on the computer.

Use the following procedure to configure the access permissions for the running class:

1. Click on the “Security” tab of the “Running Class Properties” dialog box.
2. Select “Use custom access permissions” and click on the “Edit” button. See Figure E-10.
3. If “Everyone” (or the appropriate subset of users) is not shown in the “Permissions” dialog “Name” list, click on the “Add” button to display the “Add Users or Groups” dialog box. See Figure E-11.

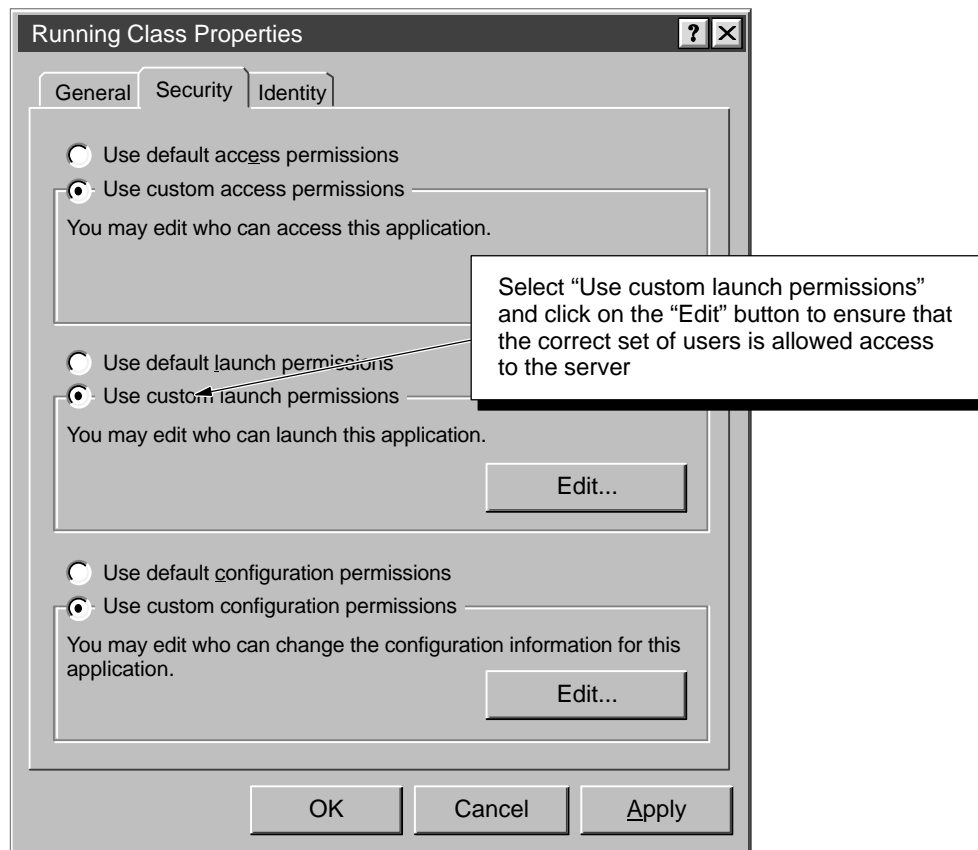


Figure E-10 Configuring the DCOM Access Permissions for the Server

4. Use the “Add Users and Groups” dialog (Figure E-11) to add users/groups as required.
5. Click on the “OK” button to return to the “Running Class Properties” dialog box.

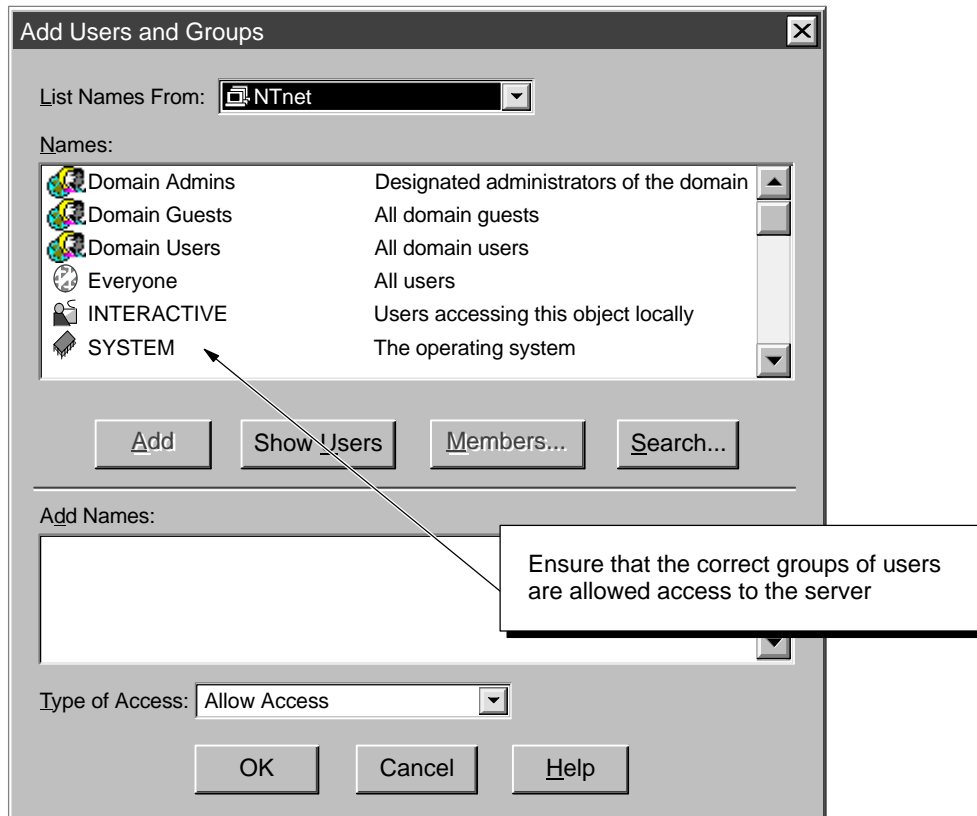


Figure E-11 Changing the Access Permissions for Users or Groups

Configuring the Identity Permissions for the Running Class

Use the following procedure to configure the identity permissions for the running class:

1. Click on the “Identity” tab and select the user to be allowed access to the server. See Figure E-12.
 - If the control engine (such as WinLC) is running as an NT service, select “This user” and enter (or browse to) the DOMAIN/LOGIN name of the user in whose security context the application will run (that is, not the remote user but the user account used to run the server application). Enter the correct password for the domain and user.
 - If the control engine is not running as an NT service, select “The interactive user” (that is, the user who is currently logged on to the computer).
2. Click on the OK button to enter the identity permissions for the running class.

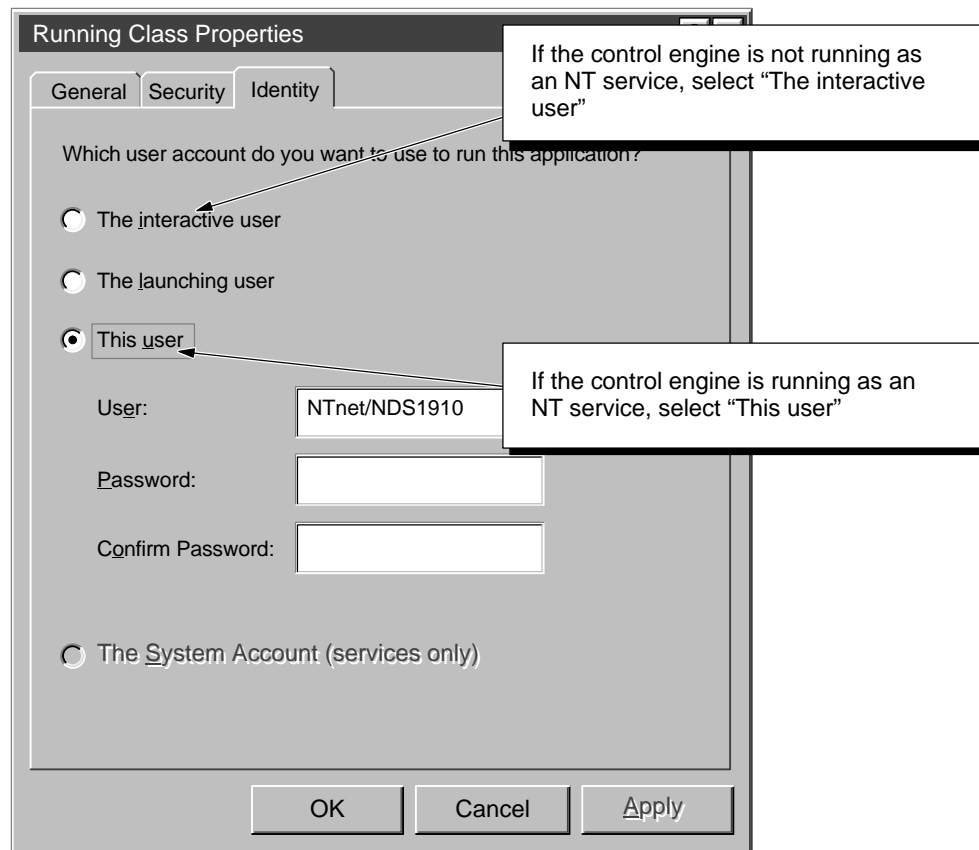


Figure E-12 Configuring the DCOM Identity Permissions for the Server

E.3 Configuring the Permissions for the Client Computer

Before you can use Computing with DCOM, you must use DCOM configuration to set application properties, such as security and location. On the computer running the client application (the application which initiates a request to a server application), you must specify the location of the server application (the application that responds to requests from a client) that will be accessed or started.

Figure E-13 lists the basic tasks required for configuring the server.

Note

You do not configure the running class properties for the client computer. You define the running class on the server computer. See Figure E-9.

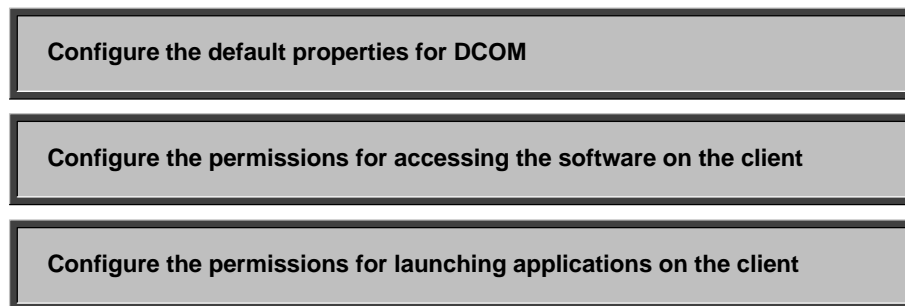


Figure E-13 Tasks for Configuring the DCOM Client



Caution

Granting permission to access applications on a computer allows other users (such as “Everyone”) to start and stop programs or to access files on your computer. Granting unlimited access to everyone on the network could cause problems from either innocent or malicious interference. Always limit access to those users who are required to use the applications or files on the computer.

Starting the DCOM Configuration Editor

To configure the DCOM client, you must run the DCOM configuration tool on the computer that will function as the client. Use the following procedure to start the DCOM configuration tool:

1. Select the **Start ► Run...** menu command from the Start menu.
2. In the “Run” dialog box, enter `dcomcnfg` and click on the “OK” button.

The DCOM configuration tool displays the “Distributed COM Configuration Properties” dialog box.

Configuring the Default Properties for DCOM Communication

Use the “Distributed COM Configuration Properties” dialog box to configure the properties of the computer for DCOM.

1. Click on the “Default Properties” tab. See Figure E-14.
2. Select the “Enable Distributed COM on this computer” option.
3. Set the “Default Authentication Level” to the “Connect” option.
4. Set the “Default Impersonation Level” to the “Identify” option.

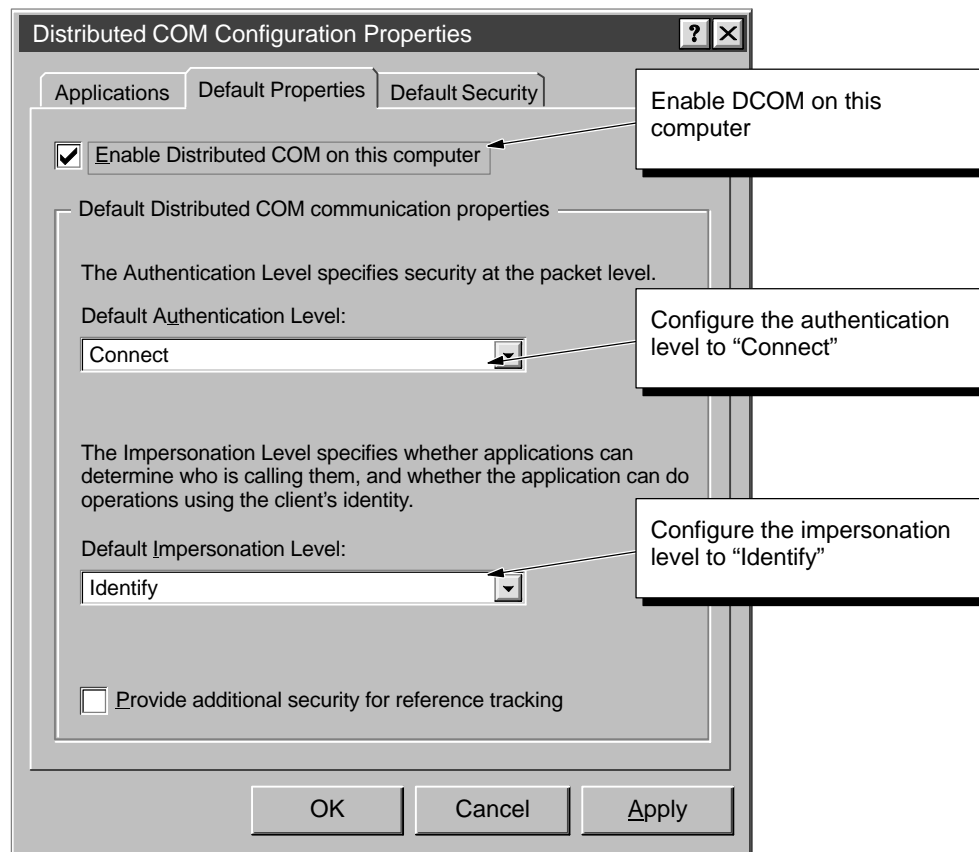


Figure E-14 Distributed COM Configuration Properties

Configuring the Permissions for Accessing Software on the Client

1. Click on the “Default Security” tab to display the security options for DCOM. See Figure E-15.
2. Click on the “Edit Default” button for “Default Access Permissions” to display the “Registry Value Permissions” dialog box.

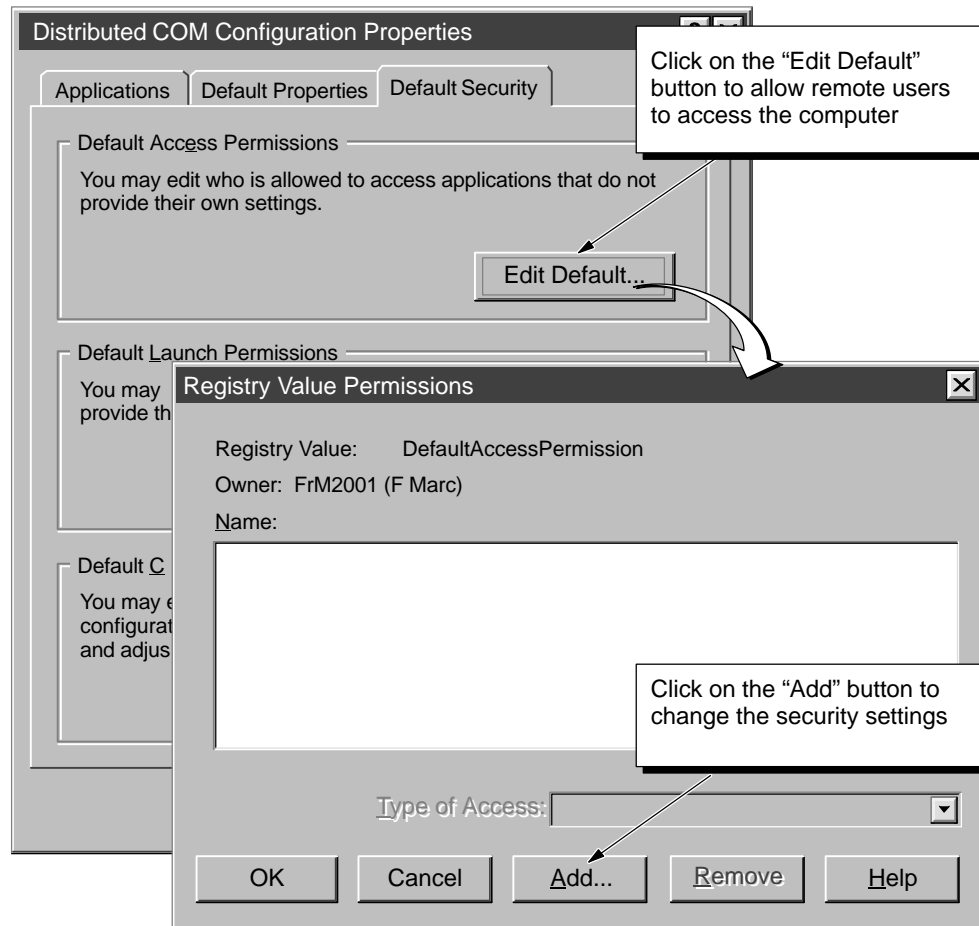


Figure E-15 Configuring the Default Access Permissions for DCOM

3. Click on the “Add” button to display the “Add Users and Groups” dialog box and change the security settings for access to the server. See Figure E-16.
4. From the “Names” field, select “Everyone” (or the appropriate subset of users) and click on the “Add” button.
5. Select “INTERACTIVE” and click on the “Add” button.
6. Select “SYSTEM” and click on the “Add” button.

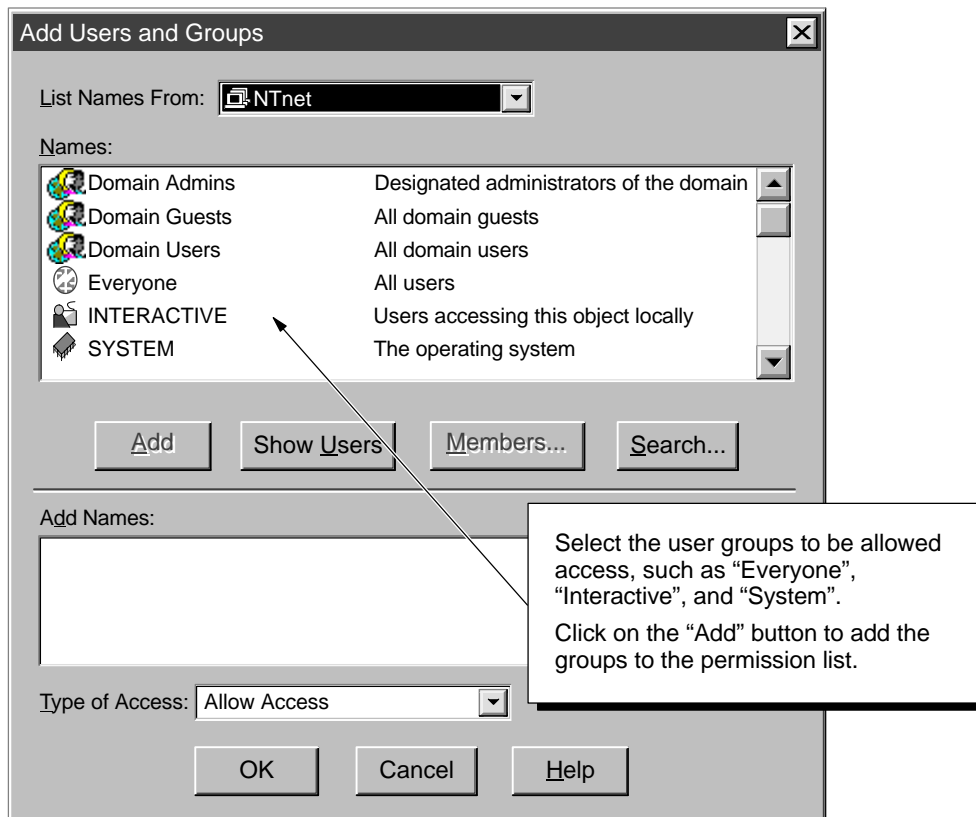


Figure E-16 Changing the Access Permissions for Users or Groups

7. Click on the "OK" button to enter these changes to the "Registry Value Permissions" dialog box.
8. Click on the "OK" button of the "Registry Value permissions" dialog box to enter the changes to the default access permissions. The "Registry Value permissions" dialog box closes and displays the "Distributed COM Configuration Properties" dialog box (Figure E-15).

**Caution**

Granting permission to access applications on a computer allows other users (such as "Everyone") to start and stop programs or to access files on your computer. Granting unlimited access to everyone on the network could cause problems from either innocent or malicious interference. Always limit access to those users who are required to use the applications or files on the computer.

Configuring the Permissions for Launching Software on the Client

1. Click on the “Edit Default” button for “Default Launch Permissions” to display the “Registry Value Permissions” dialog box. See Figure E-17.
2. Click on the “Add” button to display the “Add Users and Groups” dialog box and change the security settings for access to the server. See Figure E-18.

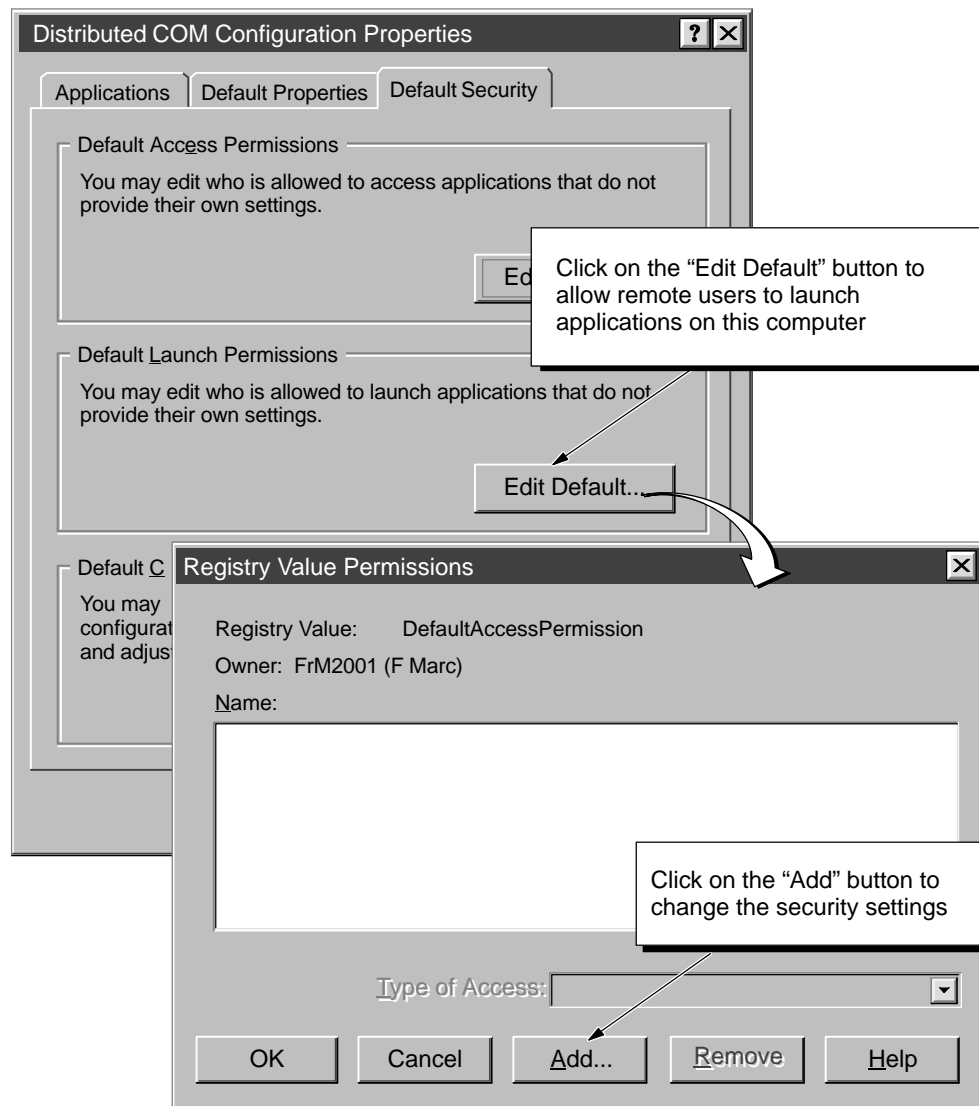


Figure E-17 Configuring the Default Launch Permissions for DCOM

3. In the “Names” field of the “Add users and Groups” dialog box (Figure E-18), select “Everyone” (or the appropriate subset of users) and click on the “Add” button.



Caution

Granting permission to access applications on a computer allows other users (such as “Everyone”) to start and stop programs or to access files on your computer. Granting unlimited access to everyone on the network could cause problems from either innocent or malicious interference. Always limit access to those users who are required to use the applications or files on the computer.

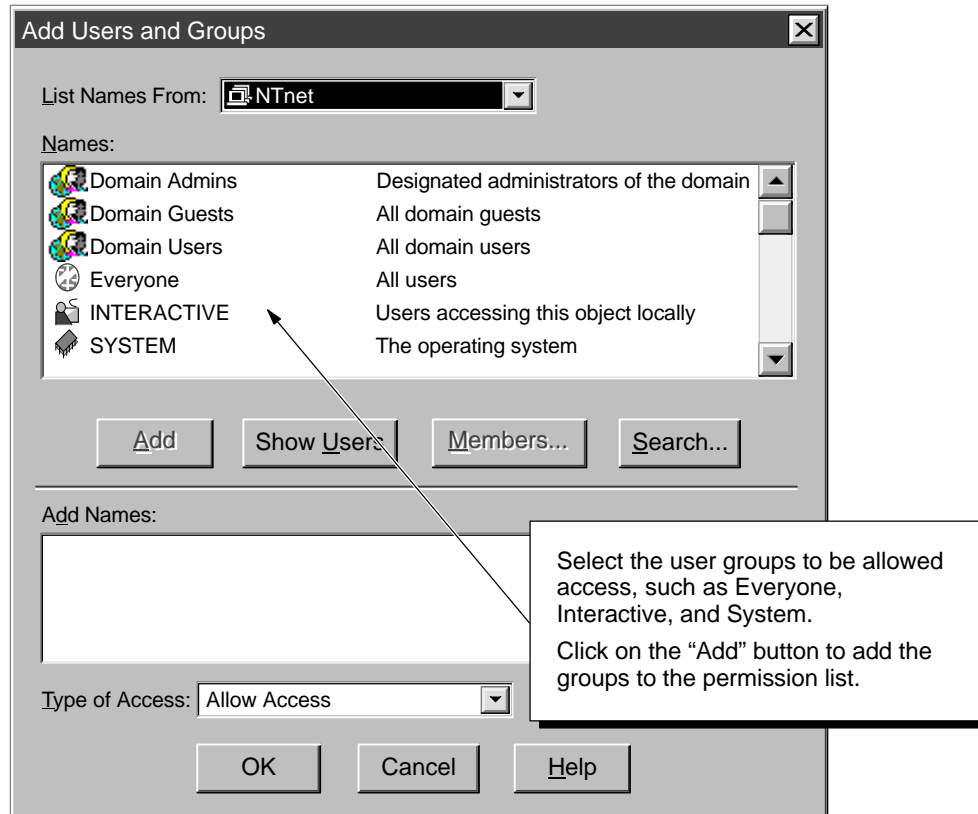


Figure E-18 Changing the Launch Permissions for Users or Groups

4. Select “INTERACTIVE” and click on the “Add” button.
5. Select “SYSTEM” and click on the “Add” button.
6. Click on the “OK” button to enter these changes to the “Registry Value Permissions” dialog box.
7. Click on the “OK” button of the “Registry Value permissions” dialog box to enter the changes to the default access permissions. The “Registry Value permissions” dialog box closes and displays the “Distributed COM Configuration Properties” dialog box.

E.4 Troubleshooting

This section provides suggestions for some of the problems that could occur with DCOM. For more information, refer to the Microsoft online product support (www.microsoft.com).

Problems with Reading and Writing Data between Two Computers over DCOM

Situation: you are running Computing on the client computer (PC1) and are connected over DCOM to WinLC on the server (PC2). You expect to read and write data between the two computers, but data updates from WinLC on PC2 do not occur.

Possible explanation: PC1 was not configured to allow PC2 to send update messages to PC1.

Possible solution:

1. Run the DCOM configuration tool (dcomcnfg) on PC1.
2. Click on the "Default Security" tab.
3. Click on the "Edit Default" button for "Default Access Permissions" to display the "Registry Value Permissions" dialog box.
4. Click on the "Add" button to display the "Add Users and Groups" dialog box and change the security settings for access to the server.
5. From the "Names" field, select "Everyone" and click on the "Add" button.
6. Click on the "OK" button to enter these changes to the "Registry Value Permissions" dialog box.
7. Click on the "OK" button to enter the changes to the default access permissions.

Guidelines for Programming with Computing

F

Chapter Overview

The SIMATIC Data control can be used not only with other SIMATIC ActiveX controls, but also with other third-party or custom ActiveX controls. To work with a custom ActiveX control, the Data control requires that the control provide a minimum of code to respond to changes in the assigned variable.

When you write programs that use the SIMATIC controls provided by the Computing software to access the control engine, be aware of the programming guidelines, especially those in regard to the use of timers in your code.

The Computing software provides a container (SoftContainer) for the SIMATIC controls and other ActiveX controls. You can also use other containers, such as Visual Basic, with the SIMATIC Controls. In order to use the SIMATIC controls in another container, the container must support the extended controls. If the container does not support these functions, you must supply program code to perform these functions.

Section	Description	Page
F.1	Guidelines for Third-Party Containers	F-2
F.2	Programming Guidelines	F-4
F.3	Guidelines for Creating Custom ActiveX Controls	F-6
F.4	Using a Custom ActiveX Control with a Data Control	F-7
F.5	Known Problems for Computing Version 3	F-10

F.1 Guidelines for Third-Party Containers

For the SIMATIC Data control to work within a third-party container, the container must support the “property browsing” functions of the Data control. To do this, the container must support the functions for “extended controls” (as defined by Microsoft for containers). An extended control is a partial control that wraps around another control to support container-specific properties, methods and events. (Refer to Microsoft’s on-line documentation for more information about containers and extended controls.)

To provide the extended control functions, the container must support the following methods:

- IOleClientSize::GetContainer
- IOleContainer::EnumObjects
- IOleControlSite::GetExtendedControl

The extended control of the container must also support a Name property.

The SoftContainer provided with the Computing software supports extended controls, as does Microsoft’s Visual Basic. Containers from other vendors (such as Borland’s Delphi version 3.0) do not support extended controls. The Siemens customer support center can help determine if your container supports the extended control functions.

If your container does not support the extended control functions, you must provide program code to perform these functions. Contact the Siemens customer support center for sample code that performs the extended control functions.

OLE Containers

Computing is an open system that may be used with OLE containers and controls from a variety of vendors. The SIMATIC controls have been tested with the following containers:

- Microsoft Visual Basic 5.0
- Microsoft Visual Basic 6.0
- Microsoft Visual Basic for Applications (VBA) for the Microsoft Office 97 applications
- Microsoft Visual C++ of Microsoft Visual Studio 5.0 and 6.0
- SoftContainer installed with the Computing software.

Some other containers from other vendors (such as Borland Delphi 3.0) do not support all the necessary ActiveX interfaces to support the “property browsing” functions of the Data control to other controls. For these containers, you must write additional code in your program to support the Microsoft “extended controls” functions for containers.

Refer to the Documentation (Especially to the List of Known Problems) for Any Third-Party OLE Container

When using the SIMATIC controls within a third-party container, please refer to the list of known problems for that container.

For example: under some conditions, Visual Basic 5.0 can cause an exception when closing. This will not affect the operation of Computing.

F.2 Programming Guidelines

The following guidelines relate specifically to Visual Basic; however, they can also apply to other programming languages.



Caution

Using the timer function improperly or using breakpoints with your subroutines that access Computing can cause problems that could potentially cause your computer or application to crash or lock up. Depending on the configuration, this could cause the application to lose communication with the control engine. This could cause unpredictable process operation, which could result in death or serious injury to personnel, and/or damage to equipment.

Always install a physical emergency stop circuit for your machine or process.

Using Timers in Your Program

The Timer function in Visual Basic version 5 allows a timer to interrupt code in progress within the same thread, which can cause problems with potentially serious consequences. If you use VB timers with Computing, observe the following guidelines:

- Always kill (disable) the timers in the Form_Unload subroutine. Otherwise, a timer can trigger an event while the VB program is shutting down; this condition could cause your computer or your application to crash, lock up, or to continue running invisibly.
- If you start your timer in the Form_Load subroutine, the timer event could occur before the other objects have finished being instantiated. In order to ensure that the objects have been properly instantiated, always start a timer in the Form_Load subroutine with a large interval (such as 1 or 2 seconds) to allow the objects to be properly instantiated. Subsequent timer intervals can be set to shorter intervals.

Using a Separate Data Control to Access Critical Data

The performance of your program can be improved by using a separate SIMATIC Data control to access frequently changing, critical data.

Disconnecting from the Control Engine

If your subroutine accesses the Data control programmatically, always disconnect from the control engine (using a Disconnect method) in the Form_Unload subroutine.

In addition, disable the AutoConnect property for the Data control if you explicitly call the Connect method within your program. This helps to ensure that the Data control does not connect unexpectedly to the control engine.

Determining the Order of AutoConnects for Multiple Data Controls

If you use multiple Data controls in your program, the order in which the different Data controls automatically connect to the control engine(s) cannot be determined. If the order in which the Data controls connect to the control engine(s) is critical, disable the AutoConnect property for the Data control and use the Connect and Disconnect methods for the individual Data controls.

F.3 Guidelines for Creating Custom ActiveX Controls

In order to create a custom ActiveX control that can be used with the SIMATIC Data control, the custom control must provide a property to which data can be written. For example, your custom control might have a Value property: when the Value property changes, then the control reacts.

Reading Data from the Data Control

If the container supports extended controls (see Section F.1), the Data control automatically finds the custom control and its properties. You use the “Properties” dialog box of the Data control to assign a variable in the control engine to the property of the custom control. (For information about assigning variables to properties, see Section 5.4.) Whenever the value of the variable in the control engine changes, the Data control updates the value of the property for the custom control.

The custom control should include a subroutine for handling the data written from the Data control. Table F-1 provides a sample subroutine for a property (Value) that reads the data written by the Data control.

Writing Data to the Data Control

For the custom control to generate (write) a change to the variable in the control engine, you must include a subroutine for handling a change in the property. Table F-1 provides a sample subroutine for writing the new value to the Data control.

Table F-1 Reading and Writing a Changed Value of a Property

Visual Basic Code
Public Property Get Value() As Long Value = Object1.Value End Property
Public Property Let Value(ByVal New_Value As Long) Object1.Value() = New_Value PropertyChanged "Value" End Property
Private Sub Value_Change() PropertyChanged "Value" End Sub

F.4 Using a Custom ActiveX Control with a Data Control

You can create a custom ActiveX control that communicates through the Data control to access the control engine. To create this sample application, you need the following items:

- Microsoft Visual Basic 5 or higher
- SIMATIC Data control from Computing
- Control engine: either WinLC or a slot PLC such as the CPU 416-2 DP ISA
- Sample program (see Section 1.1)
- STEP 7 (to download the program to the control engine and to turn on the peripheral input bits of the sample program)

You can also use the I/O Panel application to turn on the peripheral input bits of the sample program running in the control engine. See Section 1.2 for information about the I/O Panel application.

Creating a Custom ActiveX Control for Accessing the Control Engine

Use the following procedure to use a standard VB horizontal scrollbar (HScrollBar control) to create a custom ActiveX control:

1. Open a Visual Basic project for creating an ActiveX control: Use the **File ► New Project** menu command to display the “New Project” dialog box, then select the “ActiveX Control” icon (**not** the “ActiveX EXE” icon) and click on the “Open” button.
2. Add a User Control to the project: Select the **Project ► Add User Control** menu command, then select the “User Control” icon from the “Add User Control” dialog box. Clicking on the “Open” button adds the User Control to the project.
3. Select the horizontal scrollbar control (HScrollBar) in the toolbox and insert it onto the UserControl1 form.
4. Select the scrollbar control. In the Properties window, select the Max property for this control (HScroll1) and enter the following value:
255
5. Display the Code window for UserControl1 by selecting the **View ► Code** menu command. In the Code window, enter the program listed in Table F-2.
6. Close both the code window and the Object window. Visual Basic adds this ActiveX control (UserControl1) to the toolbox.

Table F-2 Sample Program for an ActiveX Control Used with Computing

Visual Basic Code
<pre>Public Property Get Value() As Integer Value = HScroll11.Value End Property</pre>
<pre>Public Property Let Value (ByVal New_Value As Integer) HScroll11.Value = New_Value PropertyChanged "Value" End Property</pre>
<pre>Public Sub HScroll11_Change() Value = HScroll11.Value End Sub</pre>

Adding the Custom Control to a Program Using the SIMATIC Data Control

1. Open a new VB project: Use the **File ► Add Project** menu command to display the "Add Project" dialog box, then select the Standard EXE icon and click on the "Open" button. Visual Basic opens a new project with an empty form in the Object window.

The Project directory area now lists two projects: Project1 contains UserControl1, and Project2 contains Form1.
2. Select the UserControl1 icon in the toolbox and insert it onto Form1 of Project2.
3. Add the Siemens SIMATIC Data control to the toolbox. For information about adding controls to the VB toolbox, see Section 1.1 and Figure 1-12.
4. Select the Data control icon in the toolbox and insert it onto Form1 of Project2.
5. Select the Data control and click the right mouse button to bring up the pop-up menu. From the pop-up menu, select **Properties** to display the "Properties" dialog box for the Data control.
6. From the "Properties" dialog box, select the "Connections" tab. Click on the "+" symbol to expand the list of controls.
7. Select the UserControl1 control and click on its "+" symbol to expand its properties list.
8. Select the Value property and enter **QB0** in the "Assigned Variable" field. See Figure 1-13. Click on the "Apply" and "OK" buttons to enter the data and close the "Properties" dialog box.

Running the Sample Program

Save the program before switching Visual Basic from Design mode to Run mode. When the sample program runs, the custom scrollbar control that you created reflects the changing value stored in QB0.

Note

If the control engine (for example, WinLC or a slot PLC such as the CPU 416-2 DP ISA) is not active, the Data control cannot make a connection. Before setting Visual Basic into Run Mode, ensure that the control engine is running.

Use the following procedure to configure the Data control for communicating with the control engine and for running the sample program.

1. Select the "Engine" tab to configure the control engine. See Figure F-1.
2. Select the "Direct Connect" option and enter either `wcs7=3` (case sensitive) for a slot PLC such as the CPU 416-2 DP ISA or `winLC` as the control engine. (See Appendix G for the correct string for other control engines.) Click on the "Apply" button to enter the data, and then click on the "OK" button to close the dialog box.
3. Switch Visual Basic from Design mode to Run mode to run the sample program.

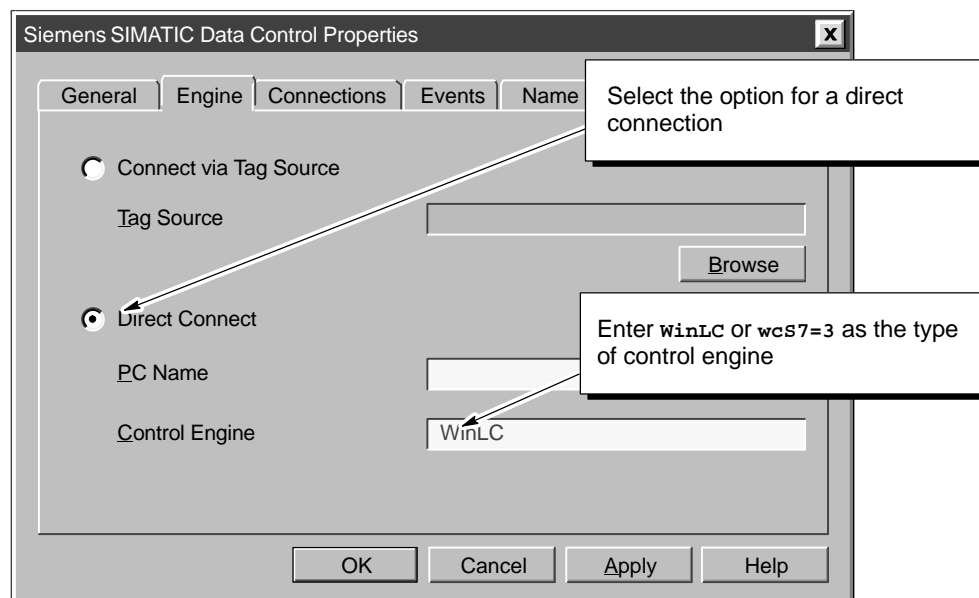


Figure F-1 Connecting to the Control Engine (Scrollbar Control Example)

F.5 Known Problems for Computing Version 3

Writing to the Peripheral Inputs

S7 control engines (PLCs) do not allow you to write to the peripheral input (PI) memory area. While Computing allows you to read data in the PI memory area, you cannot write to the PI memory area.

Reading from the Peripheral Outputs

It is not recommended that your program read from the peripheral output (PQ) memory area.

While S7 control engines (PLCs) typically allow you to write (either from STEP 7 or from your program) to the peripheral outputs (PQ memory area), the S7 control engines do not allow you to read the PQ memory areas. However, Computing allows your program to read the values for peripheral outputs. Be aware that the values read from the PQ memory area may not accurately reflect the values that you expect:

- When the control engine is in RUN mode: The values for the peripheral outputs (PQ) are typically correct. However, these values may not be correct if the corresponding I/O module has failed or is missing.
- When the control engine is in STOP mode: The values for the peripheral outputs (PQ) display the configured substitute values for safe states.

Computing will not allow you to write data to the peripheral outputs (PQ) when the control engine is in STOP mode.

Detecting the Loss of an MPI Connection

The Data control does not detect the loss of an MPI connection. Use the following procedure to detect a loss of connection:

1. Include a timer in your program.
2. At a periodic interval (such as 1 second), use the ReadVariable method of the Data control to read a specific variable (such as MB0).
3. If you receive an error message that the ReadVariable method failed, you have lost the MPI connection. Your program can then respond to the lost connection.

Handling of OPC Errors in Visual Basic

OPC methods return error codes in a HRESULT (a Long variable, in hexadecimal format). For Visual C, error conditions are handled with the HRESULT. For Visual Basic, error handling is written to the VB error object (ERR). You must add code to your VB program to access the error codes from the OPC interface.

Converting STEP 7 Time of Day (TOD) to Visual Basic vbDate

There is a problem that occurs when you read a STEP 7 variable that is stored as a TOD data type and convert that value into a Visual Basic vbDate. If you read a variable during the last half-second of the day (from 23h59m59s500ms to 23h59m59s999ms), the vbDate shows the following date for the variable: 31.12.1899 (December 31, 1899)

The time data will be correct. This aberration happens only for the last half-second of the day and is not related to the year 2000 ("Y2K").

Error Codes in Computing Version 2.0.1 and 3.0 Are Not Compatible with Version 1.2 or Version 1.1



Warning

Failing to correctly handle error conditions in your program can cause unpredictable process operation, which could result in death or serious injury to personnel, and/or damage to equipment.

Ensure that any programs written to handle errors from earlier versions of Computing (versions 1.1 or 1.2) have been updated for the error codes returned by version 2.0.1 and 3.0 of Computing. Exercise caution to ensure that you do not modify, nor permit unauthorized persons to access, any data that could cause process equipment to operate erratically. Always install a physical emergency stop circuit for your machine or process.

Many of the error codes returned by Computing have been changed for version 3.0 and 2.0.1. Programs written for earlier versions of Computing may not respond correctly to error conditions and must be updated.

Connecting to Multiple Control Engines through a Single Data Control

Connecting to multiple control engines through a single Data control causes the container to stop responding on a design-to-run transition. If you use a tag file containing multiple control engines, use a separate Data control for each one. The Data controls may share the same tag file, but each Data control must only connect controls to symbols for one control engine. This will be fixed in the next release or in a service pack.

Using Control Arrays in VB to Connect to the Control Engine

If you programmatically create a connection table (using the `ConnectObject` method in the code in your Visual Basic program to connect the objects) and then use this connection table to connect the elements of a control array to the control engine, any value changed by an element of the control array is not written automatically to the control engine. While the Data control automatically updates changes made by the control engine (by automatically reading the changed values to the elements of the control array), it does not automatically write any changed values (made with the control array) to the control engine.

- If you require that changes made with the control array utilize the “Automatic Update” option of the Data control to automatically write the change to the control engine, use the “Properties” dialog box of the Data control to create the connections for the control array (instead of writing code using the `ConnectObject` method in your VB program to make the connections).

When you use the “Properties” dialog box of the Data control to browse to the elements of the control array and assign variables in the control engine, changes made with the elements of the control array are automatically written to the control engine. (Ensure that the “Automatic Update” option for the Data control is selected.)

- If you do not require that changes made with the control array be written automatically to the control engine, you can implement code in your VB program (for example, in the code for a Button control) to write the changed value to the control object, using the `WriteVariable` method or the `WriteMultiVariables` method to manually update the value in the control engine.

Differences in the Error Codes Returned for Different PLCs

The Data control returns different error codes for different PLCs, as shown in Table F-3.

Table F-3 Error Codes

Error Condition	WinLC	CPU 416–2 DP ISA and other S7 PLCs
Write to a read-only DB	0xc0040006	0xc0040007
Wrong variable name	0xc0040008	0xc0040007
Out-of-range memory area (for example, mb40000)	0xc0040007	0x80070057

Differences Between WinLC and a Slot PLC

The following differences exist between WinLC (WinAC Basis) and a slot PLC such as the CPU 416–2 DP ISA (WinAC Pro):

- WinLC supports arrays of the the following data types: BOOL, CHAR, and STRING. MPI nodes (such as the CPU 416–2 DP ISA and other S7 controllers) do not support arrays of these data types.
- Although this practice is not recommended, WinLC allows you to read the peripheral output (PQ) memory area. Other S7 controllers (such as the CPU 416–2 DP ISA) do not allow reading peripheral outputs.

Troubleshooting: Delayed Responses of Software Using COM

Your DCOM configuration can affect local COM operations. For example, setting the “Default Authentication Level” to “None” (instead of to “Connect”) can delay the connections to software applications for up to 6 minutes as the Windows NT operating system performs its security checking. This affects not only WinAC products (such as Computing or WinLC), but other software applications that use COM (such as Microsoft Word).

When configuring your computer for DCOM, please use the entries detailed in Appendix E.

Using Control Engine Strings

Overview

Control engine strings are used in the Data Control, the Tag File Configurator, and the OPC and Diagnostic Buffer control. The control engine string identifies the location of the control engine from the STEP 7 project.

Note

From one PC, you can access only one slot PLC or one S7 network at a time.

The syntax of the control engine string depends upon the control engine type and the network being used. For example, in the formula **wcS7=xx,a,b** wcS7 is an S7 CPU on an S7 network; xx, a, and b define the location of the CPU (xx is the node address of the CPU, a is the rack number, and b is the slot number).

Note

Rack and slot number are optional, but if the rack number is specified, the slot number is mandatory. If nothing is specified, the module or CPU is accessed directly.

Control Engine Settings for WinLC

To access WinLC, enter **WinLC** into the Control Engine property.

To access WinLC through a TCP/IP LAN, you must enter the name of the PC where WinLC resides in to the PCName property (the Computer Name field of the Tag File Configurator) as one word with no spaces (for example PC_2). You do not have to set the PG/PC Interface tool for WinLC.

To access WinLC through an S7 network, see “Control Engine Settings on a SIMATIC Network” below. On an S7 network, the rack/slot address of WinLC must be 0,2.

Control Engine Settings for a Slot PLC

To access a slot PLC, such as the CPU 416–2 Dp ISA, enter **wcS7=3** into the Control Engine property.

To set the Computing interface settings for the CPU 416–2 DP ISA on the PC where the it is installed, set the PG/PC Interface to

COMPUTING—>CPU 416–2 DP ISA (local).

To access the CPU416–2 DP ISA through a TCP/IP LAN, enter the name of the PC where the CPU resides into the PCName property (the Computer Name field of the TagFile Configurator) as one word, with no spaces.

To access a slot PLC through an S7 network, see “Control Engine Settings on a SIMATIC Network” below. On an S7 network, the rack/slot address of the slot PLC must be 0,3.

Control Engine Settings on a SIMATIC Network

To access an S7 system on an MPI or PROFIBUS network, use the formula $wcS7=xx,a,b$ where xx is the node address, a is the rack number, and b is the slot number. (The longer legacy string **S7DosIntfMPI=x,a,b** is also supported.) Note that:

- The node address is always decimal.
- The rack number is 0 to 7 decimal.
- The slot number is 0 to 31 decimal.

To access an S7 system on a TCP/IP LAN, use the formula $wcIP=xxx.xxx.xxx.xxx,a,b$ where $xxx.xxx.xxx.xxx$ is the TCP/IP address, a is the rack number, and b is the slot number. Note that:

- The TCP/IP address is four decimal numbers separated by periods.
- The rack number is 0 to 7 decimal.
- The slot number is 0 to 31 decimal.

If the slot PLC is on the same computer as the Computing software, set the PG/PC Interface to **COMPUTING—><Ethernet cardname>(TCP/IP)**.

To access an S7 system on an Industrial Ethernet, use the formula $wcMAC=xx.xx.xx.xx.xx,a,b$ where $xx.xx.xx.xx.xx$ is the MAC address, a is the rack number, and b is the slot number. Note that:

- The TCP/IP address is six decimal numbers separated by periods.
- The rack number is 0 to 7 decimal.
- The slot number is 0 to 31 decimal.

If the slot PLC is on the same computer as the Computing software, set the PG/PC Interface to **COMPUTING—><Ethernet cardname>(ISO Transport)**.

If the communications card is installed on a different PC than Computing, enter the name of the PC into the PC Name Property (the Computer Name field of the Tag File Configurator), as one word with no spaces, (for example, PC_2).

Example: You wish to access an S7–315 CPU (always in slot 2) with a node address of 5.

- To access the CPU through an S7 network, the control string is **wcS7=5 (,0,0)**
- To access the CPU through an Industrial Ethernet, the control string is **wcMAC=a.b0.12.ff.3.2d,0,2**
- To access the CPU through TCP/IP LAN, the control string is **wcIP=0.0.255.255,0,2**

Note

Note that on an S7 network, the rack/slot address of WinLC is always 0,2. The rack/slot address of a slot PLC such as the CPU 4116–2 DP ISA is 0,3.

Index

A

- AboutBox method, B-1
- Absolute addresses
 - in tag file, 9-9
 - replacing symbols, 5-11
- Accessing data
 - connect/disconnect, B-4, B-8, B-9, B-14, F-5
 - separate Data control, F-4
- Accessing memory areas, SIMATIC controls, 5-1–5-12
- Accessing memory areas (S7)
 - ActiveX controls
 - Button, 6-4–6-8
 - Edit, 6-11–6-22
 - Label, 6-21–6-25
 - Slider, 6-27–6-29
 - memory areas of the S7 controllers, A-2
 - OPC controls, 2-6–2-9
- Accessing process data, 2-3–2-5
 - ActiveX controls
 - Button, 6-4–6-8
 - Edit, 6-11–6-22
 - Label, 6-21–6-25
 - Slider, 6-27–6-29
 - memory areas of S7 controllers, A-2
 - OPC controls, 2-6–2-9
 - SIMATIC controls, 5-1–5-12
- Accessing the OPC server, 2-7
- Activated property, B-1
- ActiveX controls
 - See also* ActiveX or Computing
 - Button, 6-4–6-8
 - Button control
 - description, 6-4
 - toolbar button, 6-4
 - connecting to control engine, 5-9
 - creating custom control, F-6–F-10
 - creating process form, 8-4–8-6
- ActiveX controls (continued)
 - custom controls, F-7–F-10
 - Data
 - description, 5-1
 - toolbar button, 5-1
 - Data control
 - error codes, 5-30
 - events, 5-30
 - Edit control
 - description, 6-11
 - toolbar button, 6-11
 - filtering properties, 5-13
 - Label, 6-21–6-25
 - Label control
 - description, 6-21
 - toolbar button, 6-21
 - properties
 - Button control, 6-2–6-6
 - Data control, 5-29–5-30
 - sample program
 - I/O panel, 1-4–1-10
 - Microsoft Excel, 1-15–1-19
 - other controls (VBScrollbar), 1-12–1-15
 - SoftContainer, 1-19–1-25
 - STEP 7 program, 1-3
 - sample uses, 1-2
 - sharing data among applications, 2-3, 2-4–2-6
 - Slider control, description, 6-27
 - SoftContainer
 - operating mode, 8-6–8-8
 - overview, 8-2–8-4
 - with SIMATIC controls, 5-1–5-12
- Adding connection, to Data control, 5-12
- Alignment property, B-2
- Appearance property, B-3
- ARRAY data type, A-6–A-8
- Assigning a variable, from Visual Basic, 1-6

Authorizing the Computing software, 3-2,
3-3–3-5
 procedure, 3-4
 See also README.TXT on the
 authorization disk
 guidelines, 3-4
 removing the authorization, 3-4
 running without authorization, 3-4
 transferring the authorization, 3-4
AUTHORS.EXE
 Computing authorization, 3-3–3-5
 installation (WinAC Computing), 3-5
 removing the Computing authorization, 3-4
 transferring the Computing authorization,
 3-4
AutoConnect property, B-3
AutoConnectTimeout property, B-4

B

BackColor property, B-5
bDiagBuffOK property, B-5
bEngineConnected property, B-6
BorderStyle property, B-6
BSTR, A-9
Button control, 2-4
 description, 2-4, 6-4
 events, 6-10
 Change, C-1
 Click, C-1
 Error, C-2
 KeyDown, C-3
 KeyPress, C-4
 KeyUp, C-5
 MouseDown, C-6
 MouseMove, C-7
 MouseUp, C-8
 methods, AboutBox, B-1

Button control (continued)
 properties, 6-2–6-6
 Alignment, B-2
 Appearance, B-3
 BorderStyle, B-6
 Enabled, B-19
 FalseCaption, B-21
 FalseColor, B-21
 FalsePicture, B-22
 Font, B-22
 ForeColor, B-23
 Locked, B-26
 PushButton, B-32
 StretchMode, B-38
 Style, B-39
 TrueCaption, B-40, B-42
 TrueColor, B-41
 Value, B-43
 properties and methods, 6-9
 toolbar button, 6-4

C

Caption properties, B-7
Change event, C-1
Connect method, B-7
Changing the language, D-5
CHAR data type, A-9
 use BSTR (Visual Basic), A-9
Click event, C-1
Client application (OPC), 2-3, 2-6–2-8
 connecting to Computing, 2-6–2-7
 server interfaces, 2-7
 server name, 2-7
Communicating, local and remote (DCOM),
4-1–4-7
 client and server, E-1–E-21
Component Object Model (COM)
 client and server, E-1–E-21
 local and remote, 4-1–4-7
Computer requirements, 2-3

- Computing
 - Button control
 - description, 6-4
 - properties, 6-2–6-6
 - toolbar button, 6-4
 - computer requirements, 2-3
 - configuration tool, D-6
 - Data
 - description, 5-1
 - toolbar button, 5-1
 - Data control
 - error codes, 5-30
 - events, 5-30
 - properties, 5-29–5-30
 - Edit control, description, 6-11
 - Edit control object, toolbar button, 6-11
 - error codes, Data, 5-30
 - events, Data, 5-30
 - installation
 - authorization, 3-3–3-5
 - copy-protection, 3-3–3-5
 - procedure, 3-5–3-7
 - removing the authorization, 3-4
 - system requirements, 2-3
 - transferring the authorization, 3-4
 - Label control
 - description, 6-21
 - toolbar button, 6-21
 - memory requirements, 2-3
 - OPC controls, 2-7–2-9
 - server object, 2-7
 - operating system requirements, 2-3
 - product overview, 2-1–2-8
 - properties
 - Button, 6-2–6-6
 - Data, 5-29–5-30
 - removing the authorization, 3-4
 - S7 memory areas, A-2
 - SIMATIC controls
 - Button, 6-4–6-8
 - Data control, 5-1–5-12
 - description, 2-4–2-6
 - Edit, 6-11–6-22
 - Label, 6-21–6-25
 - Slider, 6-27–6-29
 - Slider control, description, 6-27
 - SoftContainer, 8-1
 - system requirements, 2-3
 - transferring the authorization, 3-4
- Configuration tool, selecting local control engine, D-6
- Connecting to data via Data control, 2-4–2-6
- Connecting to data via OPC, 2-6–2-8
- Connection table
 - Data control, 5-16
 - sample program, 5-17
- ConnectionError event, C-1
 - error codes (Data control), 5-30
- ConnectName method, B-8
- ConnectObject method, B-9
- Control engine
 - access, 2-3–2-6
 - accessing data, 4-2–4-6
 - changing name in tag file, 9-13
 - configuring for local access, 9-11
 - configuring for remote access, 9-12
 - connecting ActiveX controls, 5-9
 - connecting over DCOM, 5-6, 9-2
 - connecting SIMATIC controls, 5-2
 - effect of scan cycle on inputs and outputs, A-2
 - identification in tag file, G-1
 - local, 4-4
 - memory areas of S7 controllers, A-2
 - OPC access, 2-6
 - OPC connection, D-2
 - OPC controls, 2-6–2-8
 - multiple, 4-6
 - remote, 4-5
 - selecting for Data control, 5-4
 - selecting local control engine, D-6
 - SIMATIC controls, 5-1–5-12
 - Button, 6-4–6-8
 - Edit, 6-11–6-22
 - Label, 6-21–6-25
 - Slider, 6-27–6-29
 - tag files, 9-1
- Control engine strings, PG/PC Interface, 5-4
- ControlEngine property, B-10
- Controller. *See* Control engine
- Copy-protection, 3-3–3-5
 - removing the authorization, 3-4
 - transferring the authorization, 3-4
- Counters, S7 memory area, A-2
- CPU
 - memory areas of S7 controllers, A-2
 - PC requirements, 2-3
- CPU 416–2 DP ISA. *See* Control engine
- Custom events, 5-15
- Customize, changing the language, D-5

D**Data, 2-4**

- accessing with Computing, 2-3–2-5

- ActiveX control objects

 - Button, 6-2–6-6

 - Data, 5-1–5-13

- memory areas of S7 controllers, A-2

- OPC controls, 2-6–2-8

- sharing among applications, 2-3–2-5

- SIMATIC controls, 5-1–5-12

 - Button, 6-4–6-8

 - Edit, 6-11–6-22

 - Label, 6-21–6-25

 - Slider, 6-27–6-29

Data control, 2-3

- adding an event, 5-15

- configuring connection properties, 5-3

- configuring for multiple control engines,
9-2–9-4

- configuring for single control engine, 5-6

- connection table, 5-16

- connections, 5-9

- containers, F-2

- custom ActiveX controls, F-6–F-10

- description, 2-4, 5-1

- error codes, 5-30–5-32

- events, 5-30

 - ConnectionError, C-1

 - ValueChanged, C-9

- methods, 5-29

 - Connect, B-7

 - ConnectName, B-8

 - ConnectObject, B-9

 - Disconnect, B-14

 - PropertyChangedName, B-30

 - PropertyChangedObject, B-31

- properties, 5-29–5-30

 - Activated, B-1

 - AutoConnect, B-3

 - AutoConnectTimeout, B-4

 - ControlEngine, B-10

 - DefaultDeadband, B-12

 - DefaultUpdateRate, B-13

Data control

- properties (continued)

 - MultipleEngines, B-27

 - PCName, B-28

 - ReadMultiVariables, B-33

 - ReadVariable, B-33

 - ShowErrorBoxes, B-36

 - TagSource, B-39

 - WriteMultiVariables method, B-44

 - WriteVariable method, B-45

- sample program

 - I/O panel, 1-4–1-10

 - Microsoft Excel, 1-15–1-19

 - other controls (VBS scrollbar), 1-12–1-15

 - SoftContainer, 1-19–1-25

 - STEP 7 program, 1-3

- selecting control engine, 5-4

- SoftContainer

 - operating mode, 8-6–8-8

 - overview, 8-2–8-4

 - process form, 8-4–8-6

- toolbar button, 5-1

- Databases, sharing data via OPC, 2-6

- DataType property, B-11

- DATE data type, A-7

- DbClick event, C-2

- DBuffer control, 7-2–7-6

 - configuring, 7-4

 - description, 2-3, 7-1

 - methods, 7-7

 - properties, 7-7–7-8

- DCOM

 - client and server, E-1–E-21

 - configuration editor, E-4, E-14

 - local and remote, 4-1–4-7

 - client configuration, E-14–E-19

 - server configuration, E-4–E-13

 - troubleshooting, E-20

- DefaultDeadband property, B-12

- DefaultUpdateRate property, B-13

- Deinstall, 3-6

 - See also Uninstalling

- Deleting a connection, 5-12

- Diagnostics buffer, DBuffer control, 7-2

- Direction property, B-14

- Disconnect method, B-14

- DisplayFormatButtons property, B-15

- DisplayHelpButton property, B-15

- DisplayHelpOnEventButton property, B-16

- DisplayLowerPanel property, B-17

- DisplayUpdateButton property, B-17

- DisplayUpperPanel property, B-18

DisplayValue property, B-18
 Distributed applications (DCOM)
 configuring server and client, E-1–E-21
 local and remote, 4-1–4-7
 Distributed Component Object Model (DCOM)
 See also DCOM
 client and server, E-1–E-21
 local and remote, 4-1–4-7

E

Edit control, 2-4
 description, 2-4, 6-11
 error codes, 6-20
 events, 6-19
 Change, C-1
 Click, C-1
 DblClick, C-2
 Error, C-2
 KeyDown, C-3
 KeyPress, C-4
 KeyUp, C-5
 MouseDown, C-6
 MouseMove, C-7
 MouseUp, C-8
 methods, AboutBox, B-1
 properties
 Alignment, B-2
 Appearance, B-3
 BackColor, B-5
 BorderStyle, B-6
 DisplayValue, B-18
 Enabled, B-19
 Factor, B-20
 Font, B-22
 ForeColor, B-23
 Locked, B-26
 Max and Min, B-26
 Offset, B-27
 Precision, B-30
 RawMax, B-32
 RawMin, B-32
 ScaleMode, B-34
 Text, B-40
 Value, B-43
 WriteMode, B-43
 WriteNow method, B-44
 Zeropad, B-45
 properties and methods, 6-18
 toolbar button, 6-11
 Effect of S7 scan cycle on inputs and outputs, A-2

Emergency stop circuit, 1-1, 2-1, 8-6
 Enabled property, B-19
 EnableSort property, B-19
 English, changing to, D-5
 Error codes
 Data control, 5-30
 Edit control, 6-20
 Error event, C-2
 Event table, sample program, 5-18
 Events

 adding to Data control, 5-15
 Button control, 6-10
 Change, C-1
 Click, C-1
 ConnectionError, C-1
 Data control, 5-30
 DblClick, C-2
 DBuffer control, 2-3, 7-1
 Edit control, 6-19
 Error, C-2
 KeyDown, C-3
 KeyPress, C-4
 KeyUp, C-5
 Label control, 6-26
 MouseDown, C-6
 MouseMove, C-7
 MouseUp, C-8
 sample program, 5-19–5-22
 Slider control, 6-35
 ValueChanged, C-9

Example

 connection table program, 5-17
 event response program, 5-19–5-22
 event table program, 5-18
 read/write Boolean data, 5-28
 read/write data, 5-23
 sample program, 1-2

Examples

 custom ActiveX control, F-7–F-10
 I/O panel, 1-4–1-10
 Microsoft Excel, 1-15–1-19
 other controls (VBS scrollbar), 1-12–1-15
 read/write with Data control, F-6
 SoftContainer, 1-19–1-25
 STEP 7 program, 1-3

F

Factor property, B-20
 FalseCaption property, B-21
 FalseColor property, B-21
 FalsePicture property, B-22

Font property, B-22
 ForeColor property, B-23
 FormatDisplay property, B-23
 French, changing to, D-5

G

German, changing to, D-5
 Guidelines
 accessing PI and PQ memory, A-2
 accessing STRING and CHAR data (BSTR), A-9
 connect/disconnect, B-4, B-8, B-9, B-14, F-5
 containers, F-2
 custom ActiveX controls, F-7–F-10
 Data control for critical data, F-4
 effect of scan cycle on inputs and outputs, A-2
 emergency stop circuit, 1-1, 2-1, 8-6
 sample programs
 I/O panel, 1-4–1-10
 Microsoft Excel, 1-15–1-19
 other controls (VBS scrollbar), 1-12–1-15
 SoftContainer, 1-19–1-25
 STEP 7 program, 1-3
 using Visual Basic timers, A-6, F-4
 WinLC authorization, 3-3
 See *also* README.TXT on the authorization disk

I

Inputs, PI and I memory areas of S7
 controllers, A-2
 Inputs and outputs, S7 controllers, A-2
 Inputs of S7 controllers
 accessing PI memory, A-2
 effect of scan cycle on inputs and outputs, A-2

Installation

 authorization, 3-3–3-5
 copy-protection, 3-3–3-5
 removing the authorization, 3-4
 transferring the authorization, 3-4
 installation and removal, 3-5–3-7
 installing the Computing authorization, 3-4
 guidelines, 3-4
 See *also* README.TXT on the authorization disk
 removing the authorization, 3-4
 system requirements, 2-3
 transferring the authorization, 3-4
 Integrating distributed applications (DCOM)
 client and server, E-1–E-21
 local and remote, 4-1–4-7

K

KeyDown event, C-3
 KeyPress event, C-4
 KeyUp event, C-5
 KnobHeight property, B-24
 KnobPicture property, B-24
 KnobWidth property, B-25

L

Label control
 description, 6-21
 events, 6-26
 Change, C-1
 Click, C-1
 DbClick, C-2
 Error, C-2
 MouseDown, C-6
 MouseMove, C-7
 MouseUp, C-8
 methods, AboutBox, B-1

Label control (continued)
 properties
 Alignment, B-2
 Appearance, B-3
 BackColor, B-5
 BorderStyle, B-6
 Caption, B-7
 Enabled, B-19
 Font, B-22
 ForeColor, B-23
 StretchMode, B-38
 Style, B-39
 properties and methods, 6-26
 toolbar button, 6-21
 Language selection, for WinAC, D-5
 LargeChange property, B-25
 Locked property, B-26

M

Max and Min properties, B-26
 Megahertz (MHz), system requirements, 2-3
 Memory areas of S7 controllers
 OPC controls, 2-6–2-8
 reference, A-2
 SIMATIC controls, 5-1–5-12
 Button, 6-4–6-8
 Edit, 6-11–6-22
 Label, 6-21–6-25
 Slider, 6-27–6-29
 Memory bits, S7 memory area (M), A-2
 Memory requirements, 2-3
 Methods
 AboutBox, B-1
 Connect, B-7
 ConnectName, B-8
 ConnectObject, B-9
 Data control, 5-29
 DBuffer control, 7-7
 Disconnect, B-14
 examples, 5-24–5-30
 PopUpHelp, B-29
 PopUpHelpOnEvent, B-29
 PropertyChangedName, B-30
 PropertyChangedObject, B-31
 ReadMultiVariables, B-33
 ReadVariable, B-33
 S7DiagBF control, SelectEvent, B-35
 WriteMultiVariables method, B-44
 WriteNow, B-44
 WriteVariable method, B-45
 MHz, system requirements, 2-3

Monitoring and modifying data
 memory areas of S7 controllers, A-2
 OPC controls, 2-6–2-8
 SIMATIC controls, 5-1–5-12
 Button, 6-4–6-8
 Edit, 6-11–6-22
 error codes (Data), 5-30
 events (Data), 5-30
 Label, 6-21–6-25
 Slider, 6-27–6-29
 MouseDown event, C-6
 MouseMove event, C-7
 MouseUp event, C-8
 MultipleEngines, B-27

N

Name of the OPC server object, 2-7
 Network communications, local and remote,
 4-1–4-7
 client and server, E-1–E-21

O

Off-the-shelf applications, OPC controls, 2-6
 Offset property, B-27
 OLE
 See also Computing, OCX, or OPC
 OLE for Process Control. *See* OPC
 OPC controls, 2-6–2-8
 OPC specification, 2-6, 2-7
 SIMATIC controls, 5-1–5-12
 Button, 6-4
 Edit, 6-11–6-22
 Label, 6-21
 Slider, 6-27–6-29
 SoftContainer, 8-1
 OPC, 2-6–2-8
 client application, 2-3, 2-7
 group object, interfaces, 2-7
 interfaces of the group object, 2-7
 interfaces of the server object, 2-7
 name of the server object, 2-7
 OPC specification, 2-6, 2-7
 server object, 2-3
 interfaces, 2-7
 name, 2-7
 sharing data among applications, 2-3–2-5,
 2-7
 used with Computing, 2-3–2-5, 2-7
 using the Data control, 2-3

- Operating system requirements, 2-3
- Outputs, Q and PQ memory areas of S7 controllers, A-2
- Outputs of S7 controllers
 - accessing PQ memory, A-2
 - effect of scan cycle on inputs and outputs, A-2
- Overview
 - Computing, 2-3–2-5
 - OPC controls, 2-3, 2-6–2-8
 - SIMATIC controls, 2-3–2-5

P

- PCName property, B-28
- Pentium, system requirements, 2-3
- Performance
 - connect/disconnect, B-4, B-8, B-9, B-14, F-5
 - Data control for critical data, F-4
- Peripheral input and output areas of S7 controllers, reference, A-2
- Personal computer (PC), system requirements, 2-3
- PI and PQ memory areas, A-2
- Picture property, B-28
- PopUpHelp method, B-29
- PopUpHelpOnEvent method, B-29
- Precision property, B-30
- Procedures
 - accessing the OPC server object, 2-7
 - authorizing the Computing software, 3-4
 - See *also* README.TXT on the authorization disk
 - adding an authorization, 3-4
 - guidelines, 3-4
 - authorizing the software, removing an authorization, 3-6
 - authorizing the WinLC software, removing an authorization, 3-4
 - installing the Computing software, 3-5
 - removing the authorization, 3-4
 - uninstalling the software, 3-6
- Process data
 - accessing, 2-3–2-5
 - OPC, 2-6–2-8
 - SIMATIC controls, 5-1–5-12
 - Button, 6-4–6-8
 - Data, 5-1–5-13
 - Edit, 6-11–6-22
 - Label, 6-21–6-25
 - Slider, 6-27–6-29

- Processor (CPU), PC requirements, 2-3
- Product overview, 2-3–2-5
 - OPC (Ole for Process Control), 2-6–2-8
- ProgID, 2-7
- Programmable Logic Controller (PLC). See Control engine
- Programmatic identifier, 2-7
- Programming
 - connect/disconnect, B-4, B-8, B-9, B-14, F-5
 - container guidelines, F-2
 - custom ActiveX controls, F-7–F-10
 - Data control for critical data, F-4
 - S7 data types in VB and C, A-5
 - sample programs
 - I/O panel, 1-4–1-10
 - Microsoft Excel, 1-15–1-19
 - other controls (VBS scrollbar), 1-12–1-15
 - SoftContainer, 1-19–1-25
 - STEP 7 program, 1-3
 - timers, F-4
- Properties
 - AboutBox method, B-1
 - Activated, B-1
 - Alignment, B-2
 - Appearance, B-3
 - AutoConnect, B-3
 - AutoConnectTimeout, B-4
 - BackColor, B-5
 - bDiagBuffOK, B-5
 - bEngineConnected, B-6
 - BorderStyle, B-6
 - Caption, B-7
 - Connect method, B-7
 - ConnectName method, B-8
 - ConnectObject method, B-9
 - ControlEngine, B-10
 - DataType, B-11
 - DefaultDeadband, B-12
 - DefaultUpdateRate, B-13
 - Direction, B-14
 - Disconnect method, B-14
 - DisplayFormatButtons, B-15
 - DisplayHelpButton, B-15
 - DisplayHelpOnEventButton, B-16
 - DisplayLowerPanel, B-17
 - DisplayUpdateButton, B-17
 - DisplayUpperPanel, B-18
 - DisplayValue, B-18
 - Enabled, B-19
 - EnableSort, B-19
 - Factor, B-20

Properties

- FalseCaption, B-21
- FalseColor, B-21
- FalsePicture, B-22
- FomatDisplay, B-23
- Font, B-22
- ForeColor, B-23
- KnobHeight, B-24
- KnobPicture, B-24
- KnobWidth, B-25
- LargeChange, B-25
- Locked, B-26
- Max and Min, B-26
- MultipleEngines, B-27
- Offset, B-27
- PCName, B-28
- Picture, B-28
- PopUpHelp method, B-29
- PopUpHelpOnEvent method, B-29
- Precision, B-30
- PropertyChangedName method, B-30
- PropertyChangedObject method, B-31
- PushButton, B-32
- RawMax, B-32
- RawMin, B-32
- ReadMultiVariables method, B-33
- ReadVariable method, B-33
- ScaleMode, B-34
- SelectEvent method, B-35
- ShowErrorBoxes, B-36
- ShowMinMax, B-36
- SIMATIC control properties
 - Button, 6-2–6-6
 - Data control, 5-29–5-30
 - DBuffer control, 7-7–7-8
- SmallChange, B-37
- StretchMode, B-38
- Style, B-39
- TagSource, B-39
- Text, B-40
- Ticks, B-40
- TrueCaption, B-40, B-42
- TrueColor, B-41
- Update method, B-42
- Value, B-43
- WriteMode, B-43
- WriteMultiVariables method, B-44
- WriteNow method, B-44
- WriteVariable method, B-45
- Zeropad, B-45

Properties and methods

- Button control, 6-9
- Edit control, 6-18
- Label control, 6-26
- Slider control, 6-34
- PropertyChangedName method, B-30
- PropertyChangedObject method, B-31
- PushButton property, B-32

R

- RAM, system requirements, 2-3
- RawMax property, B-32
- RawMin property, B-32
- Readme file, guidelines for WinLC
 - authorization, 3-3
- ReadMultiVariables method, B-33
- ReadVariable method, B-33
- Removing the Computing authorization,
 - 3-3–3-5
- Removing the Computing software, 3-6
- Removing the WinLC authorization, guidelines,
 - 3-3
 - See also* README.TXT on the authorization disk
- Requirements, computer, 2-3
- Design mode, 8-7
 - SoftContainer, 8-6–8-8
- Run mode, SoftContainer, 8-6–8-8
- Run mode (SoftContainer), appearance of the
 - Data control, 5-1

S

- S5TIME data type, A-8
- S7 controllers
 - memory areas, A-2
 - OPC controls, 2-6–2-8
 - scan cycle, A-2
 - SIMATIC controls
 - Button, 6-4–6-8
 - Edit, 6-11–6-22
 - Label, 6-21–6-25
 - Slider, 6-27–6-29

- S7 data types
 - ARRAY, A-6–A-8
 - DATE, A-7
 - in Visual Basic and C, A-5
 - S5TIME, A-8
 - STRING, A-9
 - TIME, A-10
 - TIME_OF_DAY, A-10
- S7DiagBF control
 - events, Click, C-1
 - properties
 - bDiagBuffOK, B-5
 - bEngineConnected, B-6
 - ControlEngine, B-10
 - DisplayFormatButtons, B-15
 - DisplayHelpButton, B-15
 - DisplayHelpOnEventButton, B-16
 - DisplayLowerPanel, B-17
 - DisplayUpdateButton, B-17
 - DisplayUpperPanel, B-18
 - EnableSort, B-19
 - FormatDisplay, B-23
 - PopUpHelp method, B-29
 - PopUpHelpOnEvent method, B-29
 - SelectEvent method, B-35
 - Update method, B-42
- Sample programs
 - custom ActiveX control, F-7–F-10
 - I/O panel, 1-4–1-10
 - Microsoft Excel, 1-15–1-19
 - other controls (VBS scrollbar), 1-12–1-15
 - read/write with Data control, F-6
 - SoftContainer, 1-19–1-25
 - STEP 7 program, 1-3
- ScaleMode property, B-34
- Scan cycle of S7 controllers, A-2
- SelectEvent method, B-35
- Client configuration (DCOM), E-14–E-19
- Server configuration (DCOM), E-4–E-13
- Server object (OPC), 2-3, 2-6–2-7
 - interfaces, 2-7
 - server name, 2-7
- Setting the language, D-5
- Setup program
 - authorization, 3-3–3-4
 - memory requirements, 2-3
- Sharing data among applications
 - OPC controls, 2-3–2-5, 2-6–2-8
 - OPC specification, 2-6, 2-7
 - SIMATIC controls, 2-3, 5-1–5-12
- ShowErrorBoxes property, B-36
- ShowMinMax property, B-36
- SIMATIC controls
 - Button, 6-4–6-8
 - Button control
 - description, 6-4
 - toolbar button, 6-4
 - Data
 - description, 5-1
 - toolbar button, 5-1
 - Data control, 5-1
 - DBuffer, description, 2-3, 7-1
 - DBuffer control, 2-3, 7-1
 - Edit, 6-11–6-22
 - Edit control
 - description, 6-11
 - toolbar button, 6-11
 - Label, 6-21–6-25
 - Label control
 - description, 6-21
 - toolbar button, 6-21
 - properties, Activated, B-1
 - sharing data among applications, 2-3
 - Slider, 6-27–6-29
 - Slider control, description, 6-27
 - used with Computing, 2-3
- Slider control, 2-4
 - description, 2-4, 6-27
 - events, 6-35
 - Change, C-1
 - Click, C-1
 - DbClick, C-2
 - Error, C-2
 - KeyDown, C-3
 - KeyPress, C-4
 - KeyUp, C-5
 - MouseDown, C-6
 - MouseMove, C-7
 - MouseUp, C-8

Slider control (continued)
 methods, AboutBox, B-1
 properties
 BackColor, B-5
 Direction, B-14
 DisplayValue, B-18
 Enabled, B-19
 Factor, B-20
 ForeColor, B-23
 KnobHeight, B-24
 KnobPicture, B-24
 KnobWidth, B-25
 LargeChange, B-25
 Locked, B-26
 Max and Min, B-26
 Offset, B-27
 Picture, B-28
 RawMax, B-32
 RawMin, B-32
 ScaleMode, B-34
 ShowMinMax, B-36
 SmallChange, B-37
 StretchMode, B-38
 Style, B-39
 Ticks, B-40
 Value, B-43
 properties and methods, 6-34
 Slot PLC. *See* Control engine
 SmallChange property, B-37
 SoftContainer, 8-1–8-7
 creating a process form, 8-2–8-4
 icons, 8-2–8-4
 operating mode, 8-6–8-8
 overview, 8-2–8-4
 process form, 8-4–8-6
 sample program, 1-19–1-25
 toolbars, 8-2–8-4
 Software installation
 Computing authorization, 3-3–3-5
 installing and uninstalling, 3-5–3-7
 removing the Computing authorization, 3-3–3-5
 transferring the Computing authorization, 3-3–3-5
 Software PLC. *See* Control engine
 Specifications
 OLE for Process Control, 2-6, 2-7
 system requirements, 2-3
 Spreadsheets, sharing data via OPC, 2-6
 StretchMode property, B-38
 STRING data type, A-9
 use BSTR (Visual Basic), A-9

Style property, B-39
 Switching SoftContainer modes, 8-7
 System requirements, 2-3

T

Tag File, and Data Control, 5-5
 Tag file
 components, 9-5
 configuring for local or remote access, 9-10
 control engine without symbols, 9-8
 creating, 9-6
 inserting program or control engine, 9-7
 multiple control engines, 4-6, 9-3
 remote control engine, 4-5
 Tag files, local and remote control engines, 4-1–4-7
 TagFile Configurator, 9-1–9-7
 using symbols, 9-5
 TagSource property, B-39
 Technical information, OLE for Process Control, 2-6, 2-7
 Text property, B-40
 Third-party ActiveX control, 2-3, 2-4–2-6, 8-5–8-7
 OPC controls, 2-6–2-8
 Third-party containers, F-2–F-4
 Ticks property, B-40
 TIME data type, A-10
 TIME_OF_DAY data type, A-10
 Timers, S7 memory area, A-2
 Transferring the Computing authorization, 3-3–3-5
 Transferring the WinLC authorization, guidelines, 3-3
 See also README.TXT on the authorization disk
 Troubleshooting
 DCOM, E-20
 no valid authorization, 3-3
 TrueCaption property, B-40, B-42
 TrueColor property, B-41

U

Uninstalling the Computing software, 3-6
 Update method, B-42

V

Value property, B-43

ValueChanged event, C-9

Visual Basic

- BSTR for STRING and CHAR data, A-9
- connect/disconnect, B-4, B-8, B-9, B-14, F-5

- container guidelines, F-2

- custom ActiveX controls, F-6–F-10
 - sample program, F-7–F-10

- Data control for critical data, F-4

- data types, A-5

- sample programs

 - I/O panel, 1-4–1-10

 - Microsoft Excel, 1-15–1-19

 - other controls (VBS scrollbar), 1-12–1-15

 - SoftContainer, 1-19–1-25

 - STEP 7 program, 1-3

- timers (guidelines), A-6, F-4

Visual Basic example, read/write data, 5-23

W

Warnings

- emergency stop circuit, 1-1, 2-1, 8-6

- Visual Basic timers, A-6, F-4

WinAC\Default, D-6

Windows Logic Controller (WinLC). See Control engine

WinLC, options, language, D-5

WriteMode property, B-43

WriteMultiVariables method, B-44

WriteNow method, B-44

WriteVariable method, B-45

Z

Zeropad property, B-45

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:

- | | |
|---|---|
| <input type="checkbox"/> Automotive | <input type="checkbox"/> Pharmaceutical |
| <input type="checkbox"/> Chemical | <input type="checkbox"/> Plastic |
| <input type="checkbox"/> Electrical Machinery | <input type="checkbox"/> Pulp and Paper |
| <input type="checkbox"/> Food | <input type="checkbox"/> Textiles |
| <input type="checkbox"/> Instrument and Control | <input type="checkbox"/> Transportation |
| <input type="checkbox"/> Non-electrical Machinery | <input type="checkbox"/> Other _____ |
| <input type="checkbox"/> Petrochemical | |



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.

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.

This image shows a full page of white paper with horizontal dashed lines, typical of primary school writing paper. The lines are evenly spaced and extend across the width of the page. There are no margins, text, or other markings present.