

SoMachine Basic Operating Guide

03/2015

EIO0000001354.04

www.schneider-electric.com

Schneider
 **Electric**

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2015 Schneider Electric. All rights reserved.

Table of Contents



	Safety Information	7
	About the Book	9
Part I	Getting Started with SoMachine Basic	15
Chapter 1	Introduction to SoMachine Basic	17
1.1	System Requirements and Supported Devices	18
	System Requirements	19
	Supported Devices	20
	Supported Programming Languages	22
1.2	SoMachine Basic User Interface Basics	23
	Creating Projects With SoMachine Basic	24
	Developing Programs With SoMachine Basic	25
	Navigating Within SoMachine Basic	26
	Operating Modes	27
Chapter 2	Starting with SoMachine Basic	29
2.1	The Start Page	30
	Introduction to the Start Page	31
	Registering the SoMachine Basic Software	32
	Projects Window	33
	Connect Window	34
	Project Templates Window	38
	Directly Downloading an Application	39
	Memory Management	40
Part II	Developing SoMachine Basic Applications	41
Chapter 3	The SoMachine Basic Window	43
3.1	Overview of the SoMachine Basic Window	44
	Toolbar Buttons	45
	Status Area	47
	System Settings	49
	Print Reports	51
Chapter 4	Properties	53
4.1	Overview of the Properties Window	54
	The Properties Window	55
	Project Properties	56

Chapter 5 Configuration	59
5.1 Overview of the Configuration Window	60
Overview of the Configuration Window	61
Building a Configuration	62
Chapter 6 Programming	63
6.1 Overview of the Programming Workspace	64
Overview of the Programming Workspace	64
6.2 Special Functions	66
Objects	67
Symbolic Addressing	68
Memory Allocation	70
Ladder/List Reversibility	71
How to Use the Source Code Examples	76
6.3 Configuring Program Behavior and Tasks	79
Application Behavior	80
Tasks and Scan Modes	83
6.4 Managing POUs	86
POUs	87
Managing POUs with Tasks	88
Managing Rungs	90
Free POUs	93
6.5 Master Task	95
Master Task Description	96
Configuring Master Task	97
6.6 Periodic Task	99
Creating Periodic Task	100
Configuring Periodic Task Scan Duration	102
6.7 Event Task	103
Overview of Event Tasks	104
Event Sources	105
Event Priorities and Queues	106
Creating Event Task	107
6.8 Using Tools	110
Program Messages	111
Animation Tables	113
Memory Objects	116
System Objects	118
I/O Objects	119

	Software Objects	120
	PTO Objects	121
	Communication Objects	122
	Search and Replace	123
	Symbol List	125
	Memory Consumption View.	129
	Rung Templates	131
6.9	Ladder Language Programming	134
	Introduction to Ladder Diagrams	135
	Programming Principles for Ladder Diagrams.	137
	Ladder Diagram Graphic Elements	138
	Comparison Blocks	144
	Operation Blocks	145
	Adding Comments	146
	Programming Best Practices	147
6.10	Instruction List Programming.	150
	Overview of Instruction List Programs.	151
	Operation of List Instructions.	153
	List Language Instructions.	154
	Using Parentheses	158
6.11	Grafcet (List) Programming	161
	Description of Grafcet (List) Programming	162
	Grafcet Program Structure	163
	How to Use Grafcet Instructions in a SoMachine Basic Program	167
6.12	Debugging in Online Mode	169
	Trace Window	170
	Modifying Values	172
	Forcing Values.	173
	Online Mode Modifications	174
Chapter 7	Commissioning.	175
7.1	Overview of the Commissioning Window	176
	Overview of the Commissioning Window	176
7.2	Managing the Connection to a Logic Controller	177
	Connecting to a Logic Controller	178
	Controller Information	182
	Managing the RTC	184

7.3	SoMachine Basic Simulator	185
	Overview of the SoMachine Basic Simulator	186
	SoMachine Basic Simulator I/O Manager Window	188
	SoMachine Basic Simulator Time Management Window	190
	Modifying Values Using SoMachine Basic Simulator.	193
	How to Use the SoMachine Basic Simulator	198
	Launching Simulation in Vijeo-Designer	199
7.4	Backing Up and Restoring Controller Memory	200
	Backing Up and Restoring Controller Memory	200
7.5	Downloading and Uploading Programs	202
	Downloading and Uploading Applications	203
	Controller Updates	205
Chapter 8	Saving Projects and Closing SoMachine Basic	207
	Saving a Project	208
	Saving a Project As a Template	209
	Closing SoMachine Basic	210
Appendices	211
Appendix A	SoMachine Basic Keyboard Shortcuts	213
	SoMachine Basic Keyboard Shortcuts	213
Glossary	219
Index	223

Safety Information



Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.



The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.



This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

WARNING

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

CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book



At a Glance

Document Scope

This guide describes how to use the SoMachine Basic software to configure, program, and commission applications for supported logic controllers.

Validity Note

The information in this manual is applicable **only** for SoMachine Basic products.

This document has been updated with the release of SoMachine Basic V1.3 SP1.

The technical characteristics of the devices described in this document also appear online. To access this information online:

Step	Action
1	Go to the Schneider Electric home page www.schneider-electric.com .
2	In the Search box type the reference of a product or the name of a product range. <ul style="list-style-type: none">● Do not include blank spaces in the reference or product range.● To get information on grouping similar modules, use asterisks (*).
3	If you entered a reference, go to the Product Datasheets search results and click on the reference that interests you. If you entered the name of a product range, go to the Product Ranges search results and click on the product range that interests you.
4	If more than one reference appears in the Products search results, click on the reference that interests you.
5	Depending on the size of your screen, you may need to scroll down to see the data sheet.
6	To save or print a data sheet as a .pdf file, click Download XXX product datasheet .

The characteristics that are presented in this manual should be the same as those characteristics that appear online. In line with our policy of constant improvement, we may revise content over time to improve clarity and accuracy. If you see a difference between the manual and online information, use the online information as your reference.

Related Documents

Title of Documentation	Reference Number
SoMachine Basic Generic Functions - Library Guide	EIO0000001474 (ENG) EIO0000001475 (FRA) EIO0000001476(GER) EIO0000001477 (SPA) EIO0000001478 (ITA) EIO0000001479 (CHS) EIO0000001480 (POR) EIO0000001481 (TUR)
Modicon M221 Logic Controller Advanced Functions - Library Guide	EIO0000002007 (ENG) EIO0000002008 (FRE) EIO0000002009(GER) EIO0000002010 (SPA) EIO0000002011 (ITA) EIO0000002012 (CHS) EIO0000002013 (TUR) EIO0000002014 (POR)
Modicon M221 Logic Controller - Programming Guide	EIO0000001360 (ENG) EIO0000001361 (FRE) EIO0000001362 (GER) EIO0000001363 (SPA) EIO0000001364 (ITA) EIO0000001365 (CHS) EIO0000001369 (TUR) EIO0000001368 (POR)
Modicon M221 Logic Controller - Hardware Guide	EIO0000001384 (ENG) EIO0000001385 (FRA) EIO0000001386 (GER) EIO0000001387 (SPA) EIO0000001388 (ITA) EIO0000001389 (CHS) EIO0000001370 (POR) EIO0000001371 (TUR)
Modicon TMC2 Cartridge - Programming Guide	EIO0000001782 (ENG) EIO0000001783 (FRA) EIO0000001784 (GER) EIO0000001785 (SPA) EIO0000001786 (ITA) EIO0000001787 (CHS) EIO0000001788 (POR) EIO0000001789 (TUR)

Title of Documentation	Reference Number
Modicon TMC2 Cartridge - Hardware Guide	EIO0000001768 (ENG) EIO0000001769 (FRE) EIO0000001770 (GER) EIO0000001771 (SPA) EIO0000001772 (ITA) EIO0000001773 (CHS) EIO0000001775 (TUR) EIO0000001774 (POR)
Modicon TM3 Expansion Modules Configuration - Programming Guide	EIO0000001396 (ENG) EIO0000001397 (FRA) EIO0000001398 (GER) EIO0000001399 (SPA) EIO0000001400 (ITA) EIO0000001401 (CHS) EIO0000001374 (POR) EIO0000001375 (TUR)
Modicon TM3 Digital I/O Modules - Hardware Guide	EIO0000001408 (ENG) EIO0000001409 (FRA) EIO0000001410 (GER) EIO0000001411 (SPA) EIO0000001412 (ITA) EIO0000001413 (CHS) EIO0000001376 (POR) EIO0000001377 (TUR)
Modicon TM3 Analog I/O Modules - Hardware Guide	EIO0000001414 (ENG) EIO0000001415 (FRA) EIO0000001416 (GER) EIO0000001417 (SPA) EIO0000001418 (ITA) EIO0000001419 (CHS) EIO0000001378 (POR) EIO0000001379 (TUR)
Modicon TM3 Expert Modules - Hardware Guide	EIO0000001420 (ENG) EIO0000001421 (FRA) EIO0000001422 (GER) EIO0000001423 (SPA) EIO0000001424 (ITA) EIO0000001425 (CHS) EIO0000001380 (POR) EIO0000001381 (TUR)

Title of Documentation	Reference Number
Modicon TM3 Safety Modules - Hardware Guide	EIO0000001831 (ENG) EIO0000001832 (FRA) EIO0000001833 (GER) EIO0000001834 (SPA) EIO0000001835 (ITA) EIO0000001836 (CHS) EIO0000001837 (POR) EIO0000001838 (TUR)
Modicon TM3 Transmitter and Receiver Modules - Hardware Guide	EIO0000001426 (ENG) EIO0000001427 (FRA) EIO0000001428 (GER) EIO0000001429 (SPA) EIO0000001430 (ITA) EIO0000001431 (CHS) EIO0000001382 (POR) EIO0000001383 (TUR)
Modicon TM2 Expansion Modules Configuration - Programming Guide	EIO0000000396 (ENG) EIO0000000397 (FRE) EIO0000000398 (GER) EIO0000000399 (SPA) EIO0000000400 (ITA) EIO0000000401 (CHS)
Modicon TM2 Digital I/O Modules - Hardware Guide	EIO0000000028 (ENG) EIO0000000029 (FRA) EIO0000000030 (GER) EIO0000000031 (SPA) EIO0000000032 (ITA) EIO0000000033 (CHS)
Modicon TM2 Analog I/O Modules - Hardware Guide	EIO0000000034 (ENG) EIO0000000035 (FRA) EIO0000000036 (GER) EIO0000000037 (SPA) EIO0000000038 (ITA) EIO0000000039 (CHS)

You can download these technical publications and other technical information from our website at www.schneider-electric.com.

Product Related Information

WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.¹
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

¹ For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Part I

Getting Started with SoMachine Basic

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Introduction to SoMachine Basic	17
2	Starting with SoMachine Basic	29

Chapter 1

Introduction to SoMachine Basic

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
1.1	System Requirements and Supported Devices	18
1.2	SoMachine Basic User Interface Basics	23

Section 1.1

System Requirements and Supported Devices

What Is in This Section?

This section contains the following topics:

Topic	Page
System Requirements	19
Supported Devices	20
Supported Programming Languages	22

System Requirements

Overview

The minimum system requirements for SoMachine Basic are:

- Intel Core 2 Duo processor or greater
- 1 GB RAM
- The 32- or 64-bit version of one of the following operating systems:
 - Microsoft Windows XP Service Pack 3
 - Microsoft Windows 7

Supported Devices

M221 Logic Controllers

For more information about module configuration, refer to the following programming and hardware guides:

Logic Controller Type	Hardware Guide	Programming Guide
M221 Logic Controllers	Modicon M221 Logic Controller Hardware Guide	Modicon M221 Logic Controller Programming Guide

TM3 Expansion Modules

For more information about module configuration, refer to the following programming and hardware guides of each expansion module type:

Expansion Module Type	Hardware Guide	Programming Guide
TM3 Digital I/O Expansion Modules	TM3 Digital I/O Expansion Modules Hardware Guide	TM3 Expansion Modules Programming Guide
TM3 Analog I/O Expansion Modules	TM3 Analog Modules Hardware Guide	
TM3 Expert I/O Expansion Modules	TM3 Expert I/O Modules Hardware Guide	
TM3 Safety Modules	TM3 Safety Modules Hardware Guide	
TM3 Transmitter and Receiver Modules	TM3 Transmitter and Receiver Modules Hardware Guide	

TM2 Expansion Modules

For more information about module configuration, refer to the programming and hardware guides of each expansion module type:

Expansion Module Type	Hardware Guide	Programming Guide
TM2 Digital I/O Modules	TM2 Digital I/O Modules Hardware Guide	TM2 Expansion Modules Programming Guide
TM2 Analog I/O Modules	TM2 Analog I/O Modules Hardware Guide	

TMC2 Cartridges

For more information about cartridge configuration, refer to the following programming and hardware guides:

Cartridge Type	Hardware Guide	Programming Guide
TMC2 Cartridges	TMC2 Cartridges Hardware Guide	TMC2 Cartridges Programming Guide

TMH2GDB Remote Graphic Display

For information about the Remote Graphic Display installation, compatibility, configuration, and operation, refer to the following guide:

Display Type	User Guide
Remote Graphic Display	TMH2GDB Remote Graphic Display User Guide

Supported Programming Languages

Overview

A programmable logic controller reads inputs, writes outputs, and solves logic based on a control program. Creating a control program for a logic controller consists of writing a series of instructions in one of the supported programming languages.

SoMachine Basic supports the following IEC-61131-3 programming languages:

- Ladder Diagram language
- Instruction List language
- Grafcet (List)

Section 1.2

SoMachine Basic User Interface Basics

What Is in This Section?

This section contains the following topics:

Topic	Page
Creating Projects With SoMachine Basic	24
Developing Programs With SoMachine Basic	25
Navigating Within SoMachine Basic	26
Operating Modes	27

Creating Projects With SoMachine Basic

Overview

SoMachine Basic is a graphical programming tool designed to make it easy to configure, develop, and commission programs for logic controllers.

Some Essential Terminology

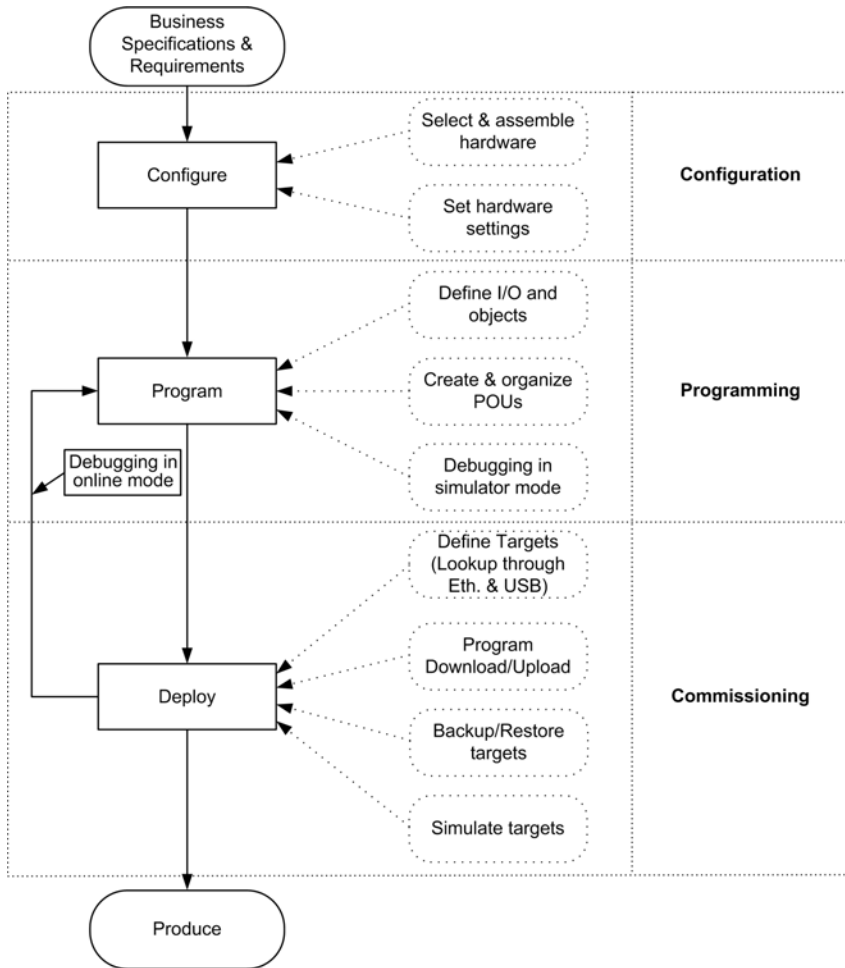
SoMachine Basic uses the following terms:

- **Project:** A SoMachine Basic project contains details about the developer and purpose of the project, the configuration of the logic controller and associated expansion modules targeted by the project, the source code of a program, symbols, comments, documentation, and all other related information.
- **Application:** Contains all parts of the project that are downloaded to the logic controller, including the compiled program, metadata, configuration information, and symbols.
- **Program:** The compiled source code that runs on the logic controller.
- **POU** (program organization unit): The reusable object that contains a variable declaration and a set of instructions used in a program.

Developing Programs With SoMachine Basic

Introduction

The following diagram shows the typical stages of developing a project in SoMachine Basic (the **Configuration**, **Programming** and **Commissioning** tabs):



Navigating Within SoMachine Basic

Start Page

The **Start Page** window is always displayed when you launch SoMachine Basic. Use this window to register your SoMachine Basic software, manage the connection to the logic controller, and create or select a project to work with.

Module Areas

Once you have selected a project to work with, SoMachine Basic displays the main window.

At the top of the main window, a toolbar ([see page 45](#)) contains icons that allow you to perform common tasks, including returning to the **Start Page** window.

Next to the toolbar, the status bar ([see page 47](#)) displays informational messages about the current state of the connection to the logic controller.

Below this, the main window is divided into a number of *modules*. Each module controls a different stage of the development cycle, and is accessible by clicking a tab at the top of the module area. To develop an application, work your way through the modules from left to right:

- **Properties** ([see page 53](#))
Set up the project properties
- **Configuration** ([see page 59](#))
Define the hardware configuration of the logic controller and associated expansion modules
- **Programming** ([see page 63](#))
Develop your program in one of the supported programming languages
- **Commissioning** ([see page 175](#))
Manage the connection between SoMachine Basic and the logic controller, upload/download applications, test, and commission the application.

Operating Modes

Introduction

The operating modes provide control to develop, debug, monitor, and modify the application when the controller is connected or not connected to SoMachine Basic.

SoMachine Basic can operate in the following modes.

- Offline mode
- Online mode
 - Simulator mode

Offline Mode

SoMachine Basic operates in offline mode when no physical connection to a logic controller has been established.

In offline mode, you configure SoMachine Basic to match the hardware components you are targeting, then develop your application.

Online Mode

SoMachine Basic operates in online mode if:

- a logic controller is physically connected to the PC.
- SoMachine Basic is simulating a virtual logic controller (known as simulator mode).

In online mode, you can proceed to download your application to the logic controller (downloading and uploading application is not possible in the simulator mode because the application is directly saved in the simulated logic controller). SoMachine Basic then synchronizes the application in the PC memory with the version stored in the logic controller, allowing you to debug, monitor, and modify the application.

You cannot modify a program in online mode.

NOTE: Online program modifications are subjected to the predefined configuration. See Memory Management ([see page 40](#)). In addition, refer to Debugging in Online Mode ([see page 169](#)) for more information.

Simulator Mode

SoMachine Basic operates in simulator mode when a connection has been established with a simulated logic controller. In simulator mode, no physical connection to a logic controller is established; instead SoMachine Basic simulates a connection to a logic controller and the expansion modules to run and test the program.

For more information, refer to SoMachine Basic Simulator ([see page 185](#)).

Chapter 2

Starting with SoMachine Basic

Section 2.1

The Start Page

What Is in This Section?

This section contains the following topics:

Topic	Page
Introduction to the Start Page	31
Registering the SoMachine Basic Software	32
Projects Window	33
Connect Window	34
Project Templates Window	38
Directly Downloading an Application	39
Memory Management	40

Introduction to the Start Page

Overview

The Start Page window is always the first window that is displayed when you start SoMachine Basic.

The Start Page window has the following windows:

- **Register** (*see page 32*)
To register SoMachine Basic software and view license details.
- **Projects** (*see page 33*)
To create a new project or open an existing project.
- **Connect** (*see page 34*)
To connect to a logic controller, download/upload application to/from the controller, back up/restore controller memory, and to flash the LEDs of the connected controller.
- **Templates** (*see page 38*)
To create a new project using an example project as a template.
- **Help**
To display the online help.
- **About**
To display information about SoMachine Basic.
- **Exit**
To exit from SoMachine Basic.

Registering the SoMachine Basic Software

Overview

You can use the SoMachine Basic software for 30 days before you are required to register the software. When you register, you receive an authorization code to use the software.

Registering your SoMachine Basic software entitles you to receive technical support and software updates.

Registering

To register your SoMachine Basic software:

Step	Action
1	Click the Register now button at the top of the Start Page window.
2	Follow the instructions on the Registration Wizard. Click the Help button for more details.

To view details on the license key installed on your PC, click **About** on the **Start Page** window.

Projects Window

Overview

Use the **Projects** window to create a new SoMachine Basic project or to open an existing SoMachine Basic, TwidoSoft, or TwidoSuite project to work with.

The right-hand area of the **Projects** window contains links to additional useful information.

Opening a SoMachine Basic Project File

Follow these steps to open a project file:

Step	Action
1	Click Projects on the Start Page window.
2	Do one of the following: <ul style="list-style-type: none"> ● Click a recent project in the Recent projects list. ● Click Create a new project. ● Click Open an existing project and select an existing SoMachine Basic project file (*.smbp) or a sample project file (*.smbe). Result: The project file opens and the Configuration tab is displayed.

Opening a TwidoSuite or TwidoSoft Project File

SoMachine Basic allows you to open applications created for Twido programmable controllers and convert them to SoMachine Basic project files.

Follow these steps to open a TwidoSuite or TwidoSoft project file:

Step	Action
1	Click Projects on the Start Page window.
2	Click Open an existing project , select any of the following in the Files of type list, and then browse and select an existing project with respective extension: <ul style="list-style-type: none"> ● TwidoSuite Project Files (*.xpr) ● Twido Archive Project Files (*.xar) ● TwidoSoft Project Files (*.twd) Result: The selected project file opens and the Configuration tab is displayed.

Connect Window

Connected Devices

The **Connect** window presents two lists of devices:

1. Local Devices

Displays all devices connected to the PC giving access to logic controllers:

- via the physical COM ports of the PC (COM1, for example)
- via USB cables
- via the virtualized COM ports (by USB-to-serial converters or Bluetooth dongles)
- via modem(s) and associated telephone number(s) that you manually add to this list

NOTE: If a COM port is selected and the **Keep Modbus driver parameters** check box is activated, the communication is established with the parameters defined in the Modbus driver.

2. Ethernet Devices

Displays all logic controllers that are accessible on the same Ethernet subnet as the PC running SoMachine Basic. Devices behind a router or any device that blocks UDP broadcasts are not listed.

The list includes logic controllers that are automatically detected by SoMachine Basic as well as any controllers that you choose to add manually.


Manually Adding Controllers

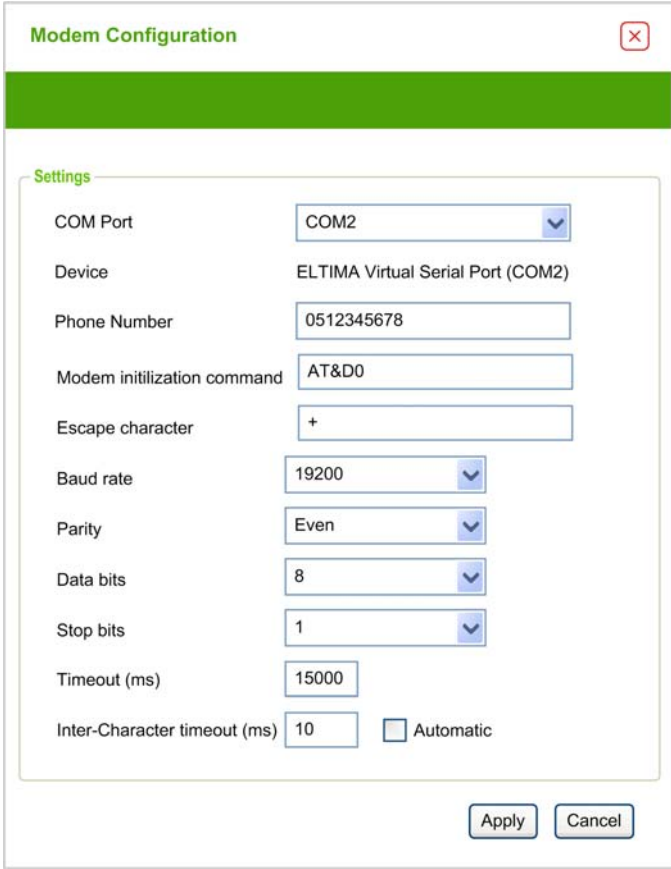

Follow these steps to add a logic controller to the **Ethernet Devices** list:

Step	Action
1	In the Remote Lookup field, type the IP address of the logic controller to add, for example, 12.123.134.21
2	Click Add to add the device to the Ethernet Devices list.

Adding Modem Connections

To add a modem connection to the **Local Devices** list:

Step	Action
1	Click  Add modem configuration button. Result: The Modem configuration window appears.

Step	Action
2	<p>Select the COM port of the modem from the drop-down list:</p> 
3	<p>Configure the communication parameters. For detailed information on the modem configuration parameters, refer to the table below.</p>
4	<p>Click Apply.</p> <p>NOTE: This button is enabled only if all settings are configured.</p> <p>Result: The modem connection is added to the Local Devices list (for example COM2@0612345678,GenericModem).</p>
5	<p>If necessary, you can edit the Modem Configuration by selecting the modem to edit in the Local Devices list and clicking on the  Modify Modem Configuration button located above the list.</p>




Modem Configuration Parameters

This table describes each parameter of the modem configuration:

Parameter	Value	Default value	Description
COM Port	COMx	-	To select the COM port of the modem from the dropdown list.
Device	-	-	Contains the modem name.
Phone Number	-	-	To enter the phone number of the modem connected to the logic controller. This text field accepts all the characters and is limited to 32 characters in total. This field must contain at least one character to be able to apply the configuration.
Modem initialization command	-	AT&D0	To edit the AT initialization command of the modem. The AT initialization command is optional (if the field is empty the AT string is sent).
Escape character	-	+	To edit the escape character for the hang-up procedure.
Baud rate	1200 2400 4800 9600 19200 38400 57600 115200	19200	To select the data transmission rate of the modem.
Parity	None Even Odd	Even	To select the parity of the transmitted data for error detection.
Data bits	7 8	8	To select the number of data bits.
Stop bits	1 2	1	To select the number of stop bits.
Timeout (ms)	0...60000	15000	To specify the transmission timeout (in ms).
Inter-Character timeout (ms)	0...10000	10	Allows you to specify the interframe timeout (in ms). If the check box Automatic is activated, the value is automatically calculated.

Connecting to a Controller

Follow these steps to connect a controller to SoMachine Basic:

Step	Action
1	Click  (Refresh Devices button) to refresh the list of connected devices.
2	Select one of the logic controllers in the Local Devices or Ethernet Devices lists. If a controller is connected by Ethernet on the same network cable as your PC, the IP address of the controller appears in the list. Selecting the IP address in the list enables  (IP Address Configuration button). Click this button to change the IP address of the controller. NOTE: If you activate the check box Write to post configuration file , the Ethernet parameters are modified in the post configuration file and kept after a power cycle.
3	If necessary, click  (Start Flashing LEDs button) to flash the LEDs of the selected controller in order to identify the controller physically. Click this button again to stop flashing the LEDs.
4	Click Login button to log in to the selected controller. If the logic controller is password protected, you are prompted to provide the password. Type the password and click OK to connect. Result: A status bar appears showing the connection progress.
5	When the connection is successfully established, details about the logic controller appear in the Selected Controller area of the window and the following buttons are available: <ul style="list-style-type: none"> ● Download application to controller: To download an application to the logic controller without opening it in SoMachine Basic. Refer to Directly Downloading an Application (see page 39). ● Memory Management: To Back up (see page 200) or restore (see page 201) the logic controller memory to or from a PC. Refer to Memory Management (see page 40). ● Upload application from controller: To create a new SoMachine Basic project file by uploading an application from the connected logic controller. Refer to Uploading an Application (see page 204).
6	Click Logout button to log out from the connected controller.

Project Templates Window

Overview

You can use example projects to form the basis of new SoMachine Basic projects.

Opening a Project Template

Follow these steps to create a new project based on a project template:

Step	Action
1	Select the Templates tab on the Start Page window.
2	Select a project template file (*.smbe) in the Projects list and click Open Template . Result: A new project is created as a copy of the selected template. NOTE: SoMachine Basic also provides a Vijeo-Designer application file and a System User Guide for some example projects. Read the description of the selected project in the Description area to know whether these files are provided with your project or not. If these files are provided, Open associated folder option gets activated on selection of such projects. Select the project and click Open associated folder to browse through the project template files (*.smbe) and Vijeo-Designer application files (*.vdz) in the Windows Explorer.

Directly Downloading an Application

Overview

You can download the application contained in a project file to a logic controller without having to open the project in SoMachine Basic. This is useful if the project is protected in *download only* mode, which prevents users from opening the project unless they have the password.

Only downloading is possible in this way. To upload an application from the logic controller to SoMachine Basic, refer to Uploading an Application (*see page 204*).

Directly Downloading an Application

To directly download an application to a logic controller:

Step	Action
1	Physically connect the PC running SoMachine Basic to the logic controller using a serial, USB, or Ethernet cable.
2	Select the Connect tab on the Start Page window.
3	Select the logic controller in the Local Devices or Ethernet Devices list and click Login . Result: SoMachine Basic establishes the connection to the logic controller.
4	Click Download application to controller .
5	In the Project File field, click the browse button, select the SoMachine Basic project file (*.smbp) to download, and click Open . Information about the selected project file appears in the Information area of the window: <ul style="list-style-type: none"> • Whether the project file is protected with a password and, if so, whether View and Download are both allowed, or Download only. • Information about the configuration contained in the project file, for example, whether the detected configuration of the logic controller system is compatible with the configuration contained in the selected project.
6	SoMachine Basic compiles the application in the selected project file. Any errors detected during compilation are listed under Compilation errors . SoMachine Basic does not allow the application to be downloaded if compilation errors have been detected; open the project in SoMachine Basic, correct the errors, then try again.
7	Before downloading, you can click the following buttons to control the current logic controller state: <ul style="list-style-type: none"> • Stop Controller • Start Controller • Initialize Controller
8	Click PC to Controller (download) . Result: SoMachine Basic downloads the application to the connected logic controller.

Memory Management

Overview

Click the **Memory Management** button on the **Connect** window to back up or restore the logic controller memory.

Select the action to perform:

- Backing up to a PC (*see page 200*)
- Restore from a PC (*see page 201*)

Part II

Developing SoMachine Basic Applications

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
3	The SoMachine Basic Window	43
4	Properties	53
5	Configuration	59
6	Programming	63
7	Commissioning	175
8	Saving Projects and Closing SoMachine Basic	207

Chapter 3

The SoMachine Basic Window

Section 3.1

Overview of the SoMachine Basic Window

What Is in This Section?

This section contains the following topics:

Topic	Page
Toolbar Buttons	45
Status Area	47
System Settings	49
Print Reports	51












Toolbar Buttons







Introduction

The toolbar appears at the top of the SoMachine Basic window to provide easy access to commonly-used functions.

Toolbar

The toolbar has the following buttons:

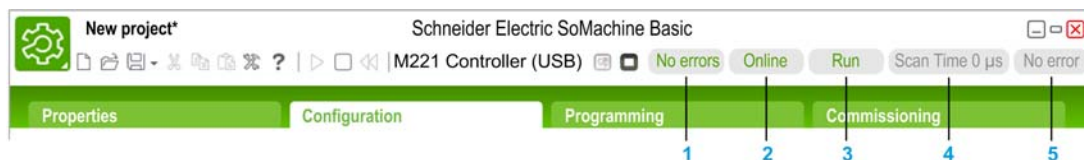
Icon	Description
	Create a new project (CTRL+N)
	Open an existing project (CTRL+O)
	Save the current project (CTRL+S). Click the down arrow to display a menu with additional save options.
	Print a report (CTRL+P). Click the down arrow to select the report to print (see page 51) or to configure the report content and format (see page 52).
	Cut (CTRL+X)
	Copy (CTRL+C)
	Paste (CTRL+V)
	Undo (CTRL+Z). Click once to undo the most recent action in the program editor. Click the down arrow and select an action from the list to undo all actions up to and including the selected action. You can undo up to 10 actions.
	Redo (CTRL+Y). Click once to cancel the most recent Undo action. Click the down arrow and select an action from the list to redo all actions up to and including the selected action. You can redo up to 10 actions.
	Display the System Settings (see page 49) window.
	Display online help (F1). For context-sensitive help, press SHIFT+F1 and click on the item you need help with.

Icon	Description
	Start the logic controller (CTRL+M). Only available in online mode and when the controller is not already in RUN state.
	Stop the logic controller (CTRL+L). Only available in online mode and when the controller is in RUN state.
	Reinitialize the logic controller. Only available in online mode.
	Compile the program.
	Log in (CTRL+G) to or log out (CTRL+H) from the selected controller. NOTE: The name of the selected controller appears to the left of this button.
	Launch (CTRL+B) or stop (CTRL+W) the SoMachine Basic simulator (<i>see page 185</i>).

Status Area

Overview

The status area at the top of the main window displays information on the present system status:



- 1 Program status:**
Indicates whether the program has errors detected or not.
- 2 Connection status:**
Indicates the connection status between SoMachine Basic and either the logic controller or the simulated logic controller.
- 3 Controller status:**
Indicates the present state of the logic controller (RUNNING, STOPPED, HALTED, and so on).
- 4 Scan time:**
Indicates the last scan time.
- 5 Controller last error detected:**
Indicates the most recent error detected. Information is extracted from the system bits and system words if the logic controller is in STOPPED or HALTED state.

Status Area Messages

The following messages can appear in the status area:

Message Type	Possible Message	Description
Program status	[No errors]	No errors detected in the program.
	[Program advisory(ies) detected]	Program is incomplete.
	[Program error(s) detected]	No program or the program contains error(s).
Connection status	[Not connected]	SoMachine Basic is running in offline mode.
	[Online]	SoMachine Basic is running in online mode.

Message Type	Possible Message	Description
Controller status (only in online mode)	[Not Connected]	Controller is not connected to SoMachine Basic.
	[Halted]	Controller is in HALTED state. Controller is stopped due to an application error being detected.
	[Stop]	Controller is in STOPPED state. Controller has a valid application which is stopped.
	[Run]	Controller is in RUNNING state. Controller is executing the application.
	[Powerless]	Controller is in POWERLESS state. Controller is powered only by the USB cable and is ready to download/upload the firmware by USB.
	[Firmware download]	Controller is downloading the firmware.
	[Firmware Error]	Firmware error detected. Version of the firmware downloading to the controller is older than present firmware version.
	[No Application]	Controller has no application.
	[Power Up]	Controller is starting (BOOTING).
Scan time (only in online mode)	[Scan Time 0 µs]	The most recent scan time in microseconds.
Controller last detected error (only in online mode)	[No error(s) detected]	No system error detected in the controller.
	[Controller could not switch to RUN mode]	Controller is not OK to run.
	[Battery level low]	Controller battery is low.
	[Run/Stop input]	Controller is stopped due to Run/Stop input command.
	[Stop command]	Controller is stopped due to stop command.
	[Software error detected (exceeding the controller scan)]	Controller is halted due to software detected error. Controller scan time overshoot. Controller scan time is greater than the period defined by the user program in configuration.
	[Stop due to detected hardware error]	Controller is stopped due to detected error in the hardware.
	[Power outage]	Controller is stopped due to power outage.
	[Controller is configured in 'Start in Stop' mode]	Controller does start in automatic application execution mode due to configuration of the startup behavior.
	[Init command]	Init in cold start.
[Unknown stop reason: {0}]	Unidentified reason.	

Refer to the programming guide of the logic controller for a complete list of the system bits and system words.

System Settings

Overview

This window allows you to set the language of the SoMachine Basic software, customize the Ladder editor, and choose the default logic controller that appears on the **Configuration** tab when you create a new project.



Changing the User Interface Language

Follow these steps to change the user interface language:

Step	Action
1	Choose System Settings → General on the System Settings window.
2	Select the language to use in the Language list. The default language is English.
3	Click Apply and close the System Settings window.
4	Close and restart SoMachine Basic to view the user interface in the new language.

Customizing the Ladder Editor

Follow these steps to customize the Ladder editor:

Step	Action
1	Choose System Settings → Ladder Editor on the System Settings window.
2	Choose the Grid lines style for the Ladder editor. <ul style="list-style-type: none"> ● Dots (default) ● Dashed Lines ● Lines
3	Set the Number of columns (11...30) for the cells in the Ladder editor. The default value of number of cells is 11. For more information, refer to Programming Principles for Ladder Diagrams (see page 137).
4	Under Tool Selection Conservation , select: <ul style="list-style-type: none"> ● Keep selected tool (default): After selecting and placing a graphic element in a rung, the most recently selected graphic element remains selected. This allows you to place the same element in a rung again without reselecting it. Press the ESC key or right-click an empty cell in the rung to select the pointer tool . ● Reset to pointer: After selecting and placing a contact or a coil in a rung, the pointer tool is automatically selected. To insert the same contact or coil element again, select it in the toolbar. 

Step	Action
5	<p>Choose the Shortcuts and toolbar style setting for the Ladder Editor:</p> <ul style="list-style-type: none">● SoMachine Basic set (default)● Asian set 1● Asian set 2● European set● American set <p>For the selected style, the table displays a list of keyboard shortcuts for each of the toolbar buttons displayed.</p>
6	Click Apply and close the System Settings window to view the changes in the Ladder editor.

Choosing a Default Logic Controller

Follow these steps to choose a default logic controller:

Step	Action
1	Choose System Settings → Configuration on the System Settings window.
2	Click Preferred controller and choose a default logic controller from the list.
3	Click Apply and close the System Settings window.
4	Close and restart SoMachine Basic to view the new default logic controller in the Configuration tab when a new project is created.

Print Reports

Presentation




You can generate customizable reports to print or to save in PDF format on the PC.

The **Print** button provides the following options:

- **Print Project Report** to print a customized report which can include the listing of the hardware components, the application architecture and the contents of the project, program, and application.
- **Print Bill Of Material** to print a listing of the hardware components used in the project configuration.
- **Settings** to customize the project report, allowing you to select which elements to include and the page layout.




Printing the Project Report

To print the project report:

Step	Action
1	Click the down arrow to the right of the Print button  on the toolbar and choose the Print Project Report menu command, or press CTRL+P. The Print Preview window is displayed.
2	<ul style="list-style-type: none"> ● Click  on the toolbar of the Print Preview window to print the project report. ● Click  on the toolbar of the Print Preview window to save the project report as a PDF file on the PC.


Printing the Bill Of Material

To print the **Bill Of Material**:

Step	Action
1	Click the down arrow to the right of the Print button  on the toolbar and choose the Print Bill Of Material menu command. The Print Preview window is displayed.
2	<ul style="list-style-type: none"> ● Click  on the toolbar of the Print Preview window to print the Bill Of Material. ● Click  on the toolbar of the Print Preview window to save the Bill Of Material as a PDF file on the PC.

Customizing the Project Report

To select which items to include in the project report and configure its layout:

Step	Action
1	Click the down arrow to the right of the Print button  on the toolbar and choose the Settings menu command. The Settings window is displayed.
2	Select the items to include in the project report: <ul style="list-style-type: none"> ● Description is the project description as in the Project Information window. ● Bill Of Material is the listing of the hardware components used in the project configuration. ● Symbols is a list of all symbols or of the symbols used in the project. ● Program is to include/exclude the following items. <ul style="list-style-type: none"> ● Behavior is the settings configured in the Behavior window. ● Application architecture is the settings configured in the Master Task and Periodic Task windows. ● POU is a Ladder language listing of all POU's in the program. ● Cross-reference is a table containing all addresses, objects, rungs, and the line of code in which they are used.
3	Click the Report node to configure the paper size and orientation.
4	Close the window.

Chapter 4

Properties

Section 4.1

Overview of the Properties Window

What Is in This Section?

This section contains the following topics:

Topic	Page
The Properties Window	55
Project Properties	56

The Properties Window

Overview

The **Properties** tab allows you to specify information about the project and whether it is to be password-protected:

- Details about the developer and the company developing the project.
- Information about the project itself.
- If the project is to be password protected, the password that must be entered correctly to open the project in SoMachine Basic.
- If the application stored in the logic controller controller is to be password protected, the password that must be entered correctly to upload the application into a SoMachine Basic project.

The screenshot shows the Schneider Electric SoMachine Basic software interface. The window title is "Schneider Electric SoMachine Basic". The interface has a green header bar with tabs for "Properties", "Configuration", "Programming", and "Commissioning". The "Properties" tab is active. On the left, a tree view shows "Project Properties" expanded, with sub-items: "Front Page", "Company", "Project Information", "Project Protection", and "Application Protection". A blue line labeled "1" points to this tree view. The main area on the right contains a form with fields for: "Last Name", "First Name", "Phone Number", "Cell Number", "Email", "Street", "City", "Zip Code" (with a "0" in the input field), "State", and "Country". A blue line labeled "2" points to the "First Name" field. At the bottom right of the form are "Apply" and "Cancel" buttons.

- 1 The left hand area displays a list of the available properties.
- 2 The right hand area displays the properties of the item that is currently selected in the left hand area.

Project Properties

Overview

Use the **Properties** window to provide details about the user of SoMachine Basic, the company developing the application, and the project. In this window, you can also password protect the project file and the application when stored in the logic controller.

Specifying Application Developer Properties

To specify the application developer properties:

Step	Action
1	Display the Properties tab and click Project Properties → Front Page .
2	Complete the information.
3	Click Apply .

NOTE: This information appears in the Windows Explorer properties window when you right-click on a SoMachine Basic project file.

Specifying Company Properties

To specify the company properties:

Step	Action
1	Display the Properties tab and click Project Properties → Company .
2	Complete the information. To upload the company logo image, click Change then browse to select the file to upload. Click Removed to delete the current image.
3	Click Apply .

Specifying Project Information

To specify project information:

Step	Action
1	Display the Properties tab and click Project Properties → Project Information .
2	Complete the information. To upload an image, such as a photograph or CAD image of the instrumented machine, click Change then browse to select the file to upload. Click Removed to delete the current image.
3	Click Apply .

Password Protecting a Project

It is possible to protect the project file. When a project is password protected, you are prompted for the password whenever the project is opened in SoMachine Basic.

Follow these steps to password protect a project file:

Step	Action
1	Display the Properties tab and click Project Properties → Project Protection .
2	Select the Active option. Required items of information are marked with an asterisk (*).
3	Type the password to use in the Password field then type it again in the Confirmation field.
4	Select one of the following options: <ul style="list-style-type: none"> ● View and Download (default): Allows you to view the contents of an application and download the application to a logic controller without knowing the password. However, you must enter the password to modify the contents of the application. ● Download Only: You can download the application to a logic controller without knowing the password. This is done through the Connect window in the Start Page (see page 34). However, you must enter the correct password when opening the project to view or modify the application.
5	Click Apply .

Removing Password Protection from a Project

Follow these steps to remove password protection from a project:

Step	Action
1	Display the Properties tab and click Project Properties → Project Protection .
2	Select the Inactive option.
3	Click Apply . NOTE: If prompted to provide the current password before the Inactive option applies successfully, type the password and click Apply .

Password Protecting an Application

SoMachine Basic allows an application stored in the logic controller to be protected with a password. This password controls uploading of the application from the logic controller into a SoMachine Basic project.

Follow these steps to password protect an application:

Step	Action
1	Display the Properties tab and click Project Properties → Application Protection .
2	Choose the level of application protection: <ul style="list-style-type: none"> ● Select Active and leave Password blank to disable application upload from the logic controller to the PC. ● Select Active and type the same password in the Password and Confirmation fields to password protect the application. You must then enter this password when prompted before uploading the application from the logic controller to the PC.
3	Click Apply .

Removing Password Protection from an Application

Follow these steps to remove password protection from an application:

Step	Action
1	Display the Properties tab and click Project Properties → Application Protection .
2	Select the Inactive option.
3	Click Apply . NOTE: If prompted to provide the current password before the Inactive option applies successfully, type the password and click Apply .

Chapter 5

Configuration

Section 5.1

Overview of the Configuration Window

What Is in This Section?

This section contains the following topics:

Topic	Page
Overview of the Configuration Window	61
Building a Configuration	62

Overview of the Configuration Window

Introduction

Use the **Configuration** window to recreate the hardware configuration of the logic controller and expansion modules to be targeted by the program.

The screenshot shows the Schneider Electric SoMachine Basic Configuration window. The window is titled "Schneider Electric SoMachine Basic" and has a "Configuration" tab selected. The window is divided into several sections:

- 1 The Hardware Tree:** A structured view of the current hardware configuration on the left side. It lists components such as "MyController (TM221M16R/G)", "Digital inputs", "Digital outputs", "Analog inputs", "High Speed Counters", "IO Bus", "Module 1 (TM3DI32K)", "Module 2 (TM2ALM3LT)", "SL1 (Serial Line)", and "SL2 (Serial Line)".
- 2 The current configuration:** A central area showing a rack of modules, including a logic controller and expansion modules.
- 3 Catalog references:** A table of supported logic controller and expansion module hardware components on the right side. The table has columns for "Reference", "Input", and "Output".
- 4 Properties:** A section at the bottom right showing the properties of the component selected in the current configuration, or the properties of the currently selected item in the Hardware Tree. It includes a "Device description" for "TM3DI32K" and a table with "5 V", "24 V", "46 mA", and "0 mA".

- 1 The Hardware Tree - a structured view of the current hardware configuration.
- 2 The current configuration - a logic controller and expansion modules.
- 3 Catalog references of all supported logic controller and expansion module hardware components. To add a component to the current hardware configuration, drag and drop it onto the current configuration.
- 4 The properties of the component selected in the current configuration, or the properties of the currently selected item in the Hardware Tree.

Building a Configuration

Replacing the Default Logic Controller

When you create a new SoMachine Basic project, a logic controller reference appears in the central area of the **Configuration** window.

Step	Action
1	Click the Configuration tab.
2	Expand the logic controller category in the catalog area on the right, if it is not already displayed.
3	Select a logic controller reference. A short description of the physical properties of the logic controller appear in the Device description area.
4	Drag the logic controller reference over the image of the existing logic controller in the central area of the window and drop it.
5	Click Yes when prompted to confirm replacing the logic controller reference.

NOTE: The default controller reference is specified in the **System Settings** window ([see page 49](#)).

Configuring the Logic Controller

Use the **Configuration** window to configure the logic controller.

Refer to the *Programming Guide* of the logic controller used in the configuration for details.

Configuring Expansion Modules

Use the **Configuration** window to add and configure expansion modules.

Refer to the *Programming Guide* of the expansion module used in the configuration for details.

Chapter 6

Programming

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Overview of the Programming Workspace	64
6.2	Special Functions	66
6.3	Configuring Program Behavior and Tasks	79
6.4	Managing POUs	86
6.5	Master Task	95
6.6	Periodic Task	99
6.7	Event Task	103
6.8	Using Tools	110
6.9	Ladder Language Programming	134
6.10	Instruction List Programming	150
6.11	Grafcet (List) Programming	161
6.12	Debugging in Online Mode	169

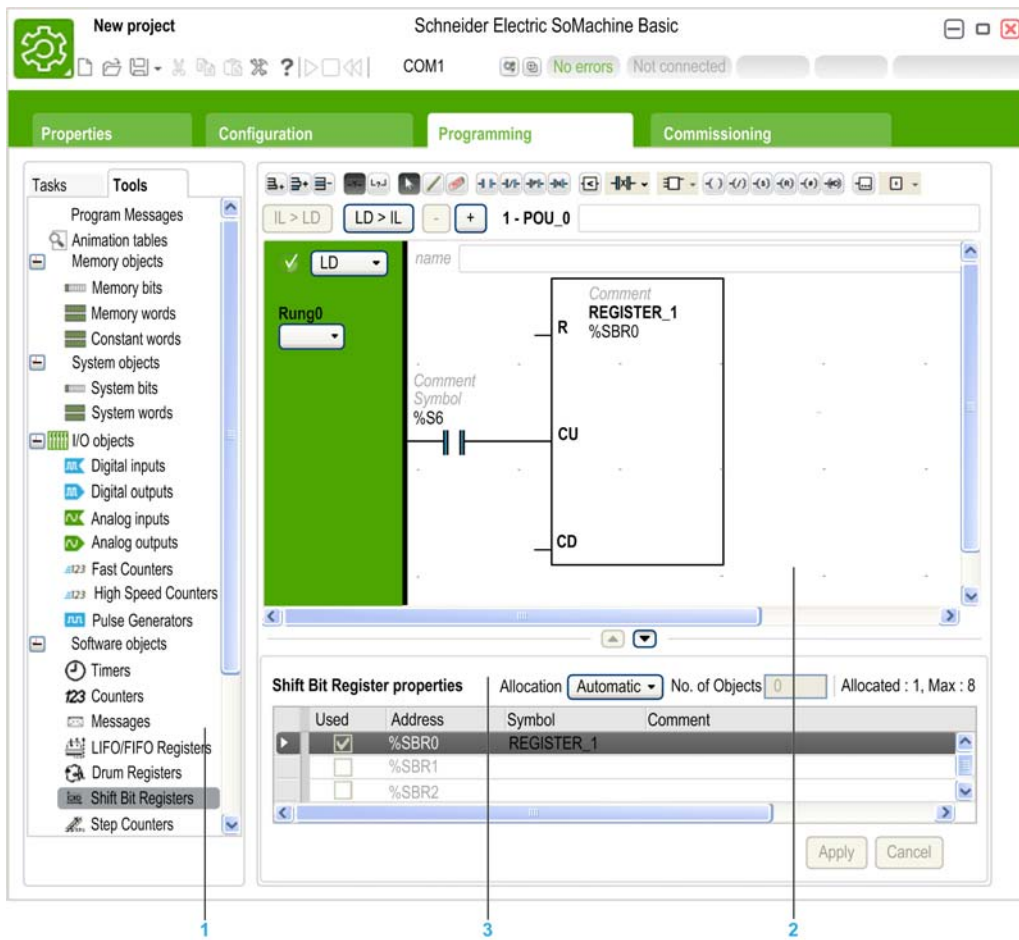
Section 6.1

Overview of the Programming Workspace

Overview of the Programming Workspace

Overview

The **Programming** tab is split into 3 main areas:



- 1 The Programming Tree allows you to configure the properties of the program and its objects, and functions, as well as a number of tools you can use to monitor and debug the program.

- 2** The upper central area is the programming workspace where you enter the source code of your program.
- 3** The lower central area allows you to view and configure the properties of the item currently selected in the program or the Programming Tree.

Section 6.2

Special Functions

What Is in This Section?

This section contains the following topics:

Topic	Page
Objects	67
Symbolic Addressing	68
Memory Allocation	70
Ladder/List Reversibility	71
How to Use the Source Code Examples	76

Objects

Overview

In SoMachine Basic, the term *object* is used to represent an area of logic controller memory reserved for use by an application. Objects can be:

- Simple software variables, such as memory bits and words
- Addresses of digital or analog inputs and outputs
- Controller-internal variables, such as system words and system bits
- Predefined system functions or function blocks, such as timers and counters.

Controller memory is either pre-allocated for certain object types, or automatically allocated when an application is downloaded to the logic controller.

Objects can only be addressed by a program once memory has been allocated. Objects are addressed using the prefix `%`. For example, `%MW12` is the address of a memory word, `%Q0.3` is the address of an embedded digital output, and `%TM0` is the address of a `Timer` function block.

Symbolic Addressing

Introduction

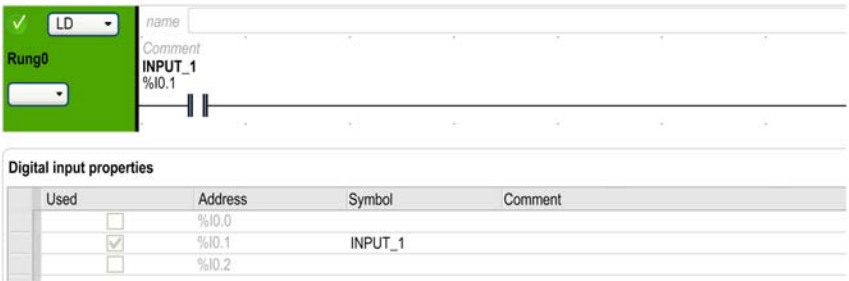
SoMachine Basic supports the symbolic addressing of language objects; that is, the indirect addressing of objects by name. Using symbols allows for quick examination and analysis of program logic, and greatly simplifies the development and testing of an application.

Example

For example, WASH_END is a symbol that could be used to identify an instance of a `Timer` function block representing the end of a wash cycle. Recalling the purpose of this name is easier than trying to remember the role of a program address such as `%TM3`.

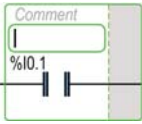
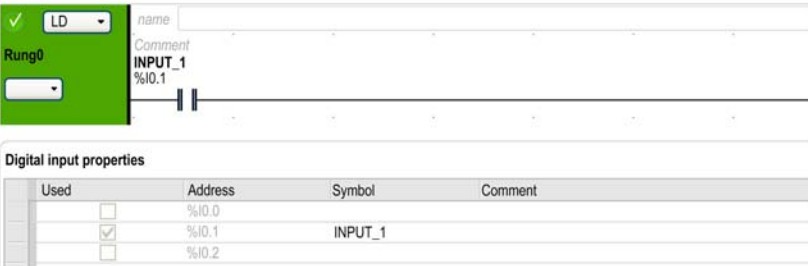
To Define a Symbol in the Properties Window

To define a symbol in the properties window:

Step	Action
1	Select the Tools tab in the left-hand area of the Programming window.
2	Select the type of object with which to define a symbol, for example I/O objects → Digital inputs , to display the properties of digital inputs. The properties window of the object type appears in the lower central area of the Programming window.
3	Double-click in the Symbol column of the properties table and type the symbol to define for a particular item, for example <code>Input_1</code> for the input <code>%I0.2</code> 
4	Click Apply .

To Define a Symbol in the Ladder Editor

To define a symbol within the Ladder editor:

Step	Action																
1	<p>In the Ladder editor, click the Symbol line of a graphic element, for example a latch or function block. A cursor appears:</p> 																
2	<p>Type the symbol to use, for example <code>Input_1</code> and press Enter. The following rules apply to symbols:</p> <ul style="list-style-type: none"> • A maximum of 32 characters. • Letters (A-Z), numbers (0-9), or underscores (_). • First character must be a letter. You cannot use the percentage sign (%). • Symbols are not case-sensitive. For example, <code>Pump1</code> and <code>PUMP1</code> are the same symbol and can only be used uniquely for any given object; that is, you cannot assign the same symbol to different objects. 																
3	<p>If the graphic element is not yet associated with an object, the Remark window appears. Select an object to associate with the symbol and click OK. Otherwise, click Yes when prompted to associate the symbol with the object.</p>																
4	<p>Double-click either the symbol or object of the graphic element to display the symbol in the Symbol column of the properties window:</p>  <table border="1"> <thead> <tr> <th>Used</th> <th>Address</th> <th>Symbol</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>%I0.0</td> <td></td> <td></td> </tr> <tr> <td><input checked="" type="checkbox"/></td> <td>%I0.1</td> <td>INPUT_1</td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%I0.2</td> <td></td> <td></td> </tr> </tbody> </table>	Used	Address	Symbol	Comment	<input type="checkbox"/>	%I0.0			<input checked="" type="checkbox"/>	%I0.1	INPUT_1		<input type="checkbox"/>	%I0.2		
Used	Address	Symbol	Comment														
<input type="checkbox"/>	%I0.0																
<input checked="" type="checkbox"/>	%I0.1	INPUT_1															
<input type="checkbox"/>	%I0.2																

Displaying Symbols in Instruction List Code

When displaying Instruction List code for a rung, select the **symbols** check box to display any defined symbols instead of direct object references in the code.

Displaying All Defined Symbols

Choose **Tools** → **Symbol list** to display a list of all defined symbols ([see page 125](#)).

Storing Symbols

Symbols are stored in the logic controller as part of a SoMachine Basic application.

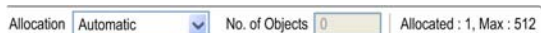
Memory Allocation

Introduction

SoMachine Basic allows you to pre-allocate (reserve) blocks of logic controller memory for use by certain object types used in a program, including simple objects (memory words, constant words) and software objects (function blocks).

Allocation Modes

In offline mode, you can specify the memory allocation mode for each object type. When configuring these objects (**Programming** → **Tools**), the following window then appears above the list of configurable objects:



Allocation Automatic No. of Objects 0 Allocated : 1, Max : 512

Choose the memory allocation mode to use:

- **Automatic.** All objects from offset 0 to the highest memory address used in the program, or associated with a symbol, are automatically allocated in logic controller memory. For example: if the memory word `%MW20` is used in the program, all objects from `%MW0` to `%MW20` inclusive (21 objects) are automatically allocated in memory. If you later switch to online mode, you cannot allocate new memory objects with addresses higher than the highest address that was used before you went online.
- **Manual.** Specify a number of objects to be allocated in memory in the **No. of Objects** box. When you switch to online mode, you can add new contacts, coils, or equations in your program (up to the limit of memory allocated) without having to log out from the logic controller, modify the program, log in, and download the application again.

SoMachine Basic displays the total number of **Allocated** memory objects and the **Maximum** number of memory objects available in the logic controller.

Ladder/List Reversibility

Introduction

SoMachine Basic supports conversion of rungs from Ladder Diagram to Instruction List and from Instruction List back to Ladder Diagram. This is called *program reversibility*.

In SoMachine Basic, you can toggle rungs between programming languages at any time as required. You can therefore display a program with some rungs in Ladder Diagram and other rungs in Instruction List.

Understanding Reversibility

A key to understanding program reversibility is examining the relationship between a Ladder Diagram rung and the associated Instruction List rung:

- **Ladder Diagram rung:** A collection of Ladder Diagram instructions that constitute a logical expression.
- **List sequence:** A collection of Instruction List programming instructions that correspond to the Ladder Diagram instructions and represents the same logical expression.

The following illustration displays a common Ladder Diagram rung and its equivalent program logic expressed as a sequence of Instruction List instructions.



Equivalent Instruction List instruction:

name			Comment
0000	LD	%I0.5	Comment
0001	OR	%I0.4	Comment
0002	ST	%Q0.4	Comment

A program is always stored internally as Instruction List instructions, regardless of whether it is originally written in the Ladder Diagram or Instruction List language. SoMachine Basic takes advantage of the program structure similarities between the 2 languages and uses this internal Instruction List image of the program to display it either as an Instruction List program, or graphically as a Ladder Diagram.

Instructions Required for Reversibility

The structure of a reversible function block in Instruction List language requires the use of the following instructions:

- **BLK** marks the block start, and defines the beginning of the rung and the start of the input portion to the block.
- **OUT_BLK** marks the beginning of the output portion of the block.
- **END_BLK** marks the end of the block and the rung.

The use of these reversible function block instructions is not mandatory for a properly functioning Instruction List program.

Programming Situations and IL/Ladder Reversibility

The following table lists programming situations for the Ladder or IL languages which, if left untreated, generate advisories or errors and a possible loss of reversibility.

Situation	IL	Ladder	Rung reversible
Jump to a label which has not been defined	Error	Error	Yes
Call to undefined subroutine	Error	Error	Yes
Activate or deactivate an undefined Grafcet step	Error	Error	Yes
Jump instruction between parentheses	Error	-	No
Label between parentheses	Error	-	No
Subroutine between parentheses	Error	-	No
More than 32 nested parentheses	Error	-	No
Closing parenthesis without opening parenthesis	Error	-	No
Reserved	-	-	-
Unbalanced parentheses	Error	-	No
BLK without END_BLK	Error	-	No
OUT_BLK or END_BLK without BLK	Error	-	No
Label definition not followed by LD or BLK	Error	-	No
Subroutine definition not followed by LD or BLK	Error	-	No
Reserved	-	-	-
More than 11 nested MPS	Error	-	No
MRD without MPS	Error	-	No
MPP without MPS	Error	-	No
Use Grafcet instruction in POST	Error	Error	Yes
Grafcet definition not followed by BLK or LD	Error	-	No
Unbalanced stack operations	Error	-	No
Reserved	-	-	-

Situation	IL	Ladder	Rung reversible
Duplicate label	Error	Error	Only LD->IL
Duplicate Subroutine	Error	Error	Only LD->IL
Duplicate Grafcet step	Error	Error	Only LD->IL
Reserved	-	-	-
Duplicate POST	Error	Error	Only LD->IL
Nested FB	Error	-	No
OUT_BLK between BLK and END_BLK	Error	-	No
BLK not followed by LD	Error	-	No
LD of FB output not in OUT_BLK	Error	-	No
FB outputs used outside their respective FB structure	Error	-	No
FB outputs repeated or out of order	Error	-	No
FB inputs not in BLK before OUT_BLK	Error	-	No
FB inputs used outside their respective FB structure	Error	-	No
FB inputs repeated or out of order	Error	-	No
Label declared in BLK	Error	-	No
Subroutines declared in BLK	Error	-	No
Grafcet steps declared in BLK	Error	-	No
Attempted LD of a non FB output in OUT_BLK	Error	-	No
FB output used between BLK and END_BLK	Error	-	No
Nested subroutines	Error	Error	No
Subroutine call between MPS and MPP	Error	Error	No
Subroutine call between parentheses	Error	-	No
Reserved	-	-	
First instruction of program not a rung delimiter	Error	-	No
Jump instruction between MPS and MPP	Error	Error	No
Rung contains syntax error	Error	-	No
Reserved	-	-	-
Reserved	-	-	-
Program instructions following JMP or END unconditional instructions	Error	-	No
Rung beginning with LD instruction does not terminate with a conditional action instruction	Advisory	-	No
Action instruction between parentheses	Error	-	No
Stack instruction between parentheses	Error	-	No

Situation	IL	Ladder	Rung reversible
Direct-access instructions for FB (ex: "CU %C0 ")	Advisory	-	No
Action instructions in the input section of a FB	Error	-	No
Instructions after END_BLK	Error	-	No
FB outputs used with AND and OR instructions	Advisory	-	No
OR instruction inside a FB output not between parentheses	Advisory	-	No
Instruction preceding MRD or MPP not either a conditional action or associated with stack instructions	Advisory	-	No
Unnested OR between MPS and MPP	Advisory	-	No
OR after an action instruction	Advisory	-	No
OR after MPS, MRD or MPP	Advisory	-	No
Reserved	-	-	
Subroutine call or JMPC not the last action instruction of the rung	Advisory	Error	No
Canonic rung exceeds 7x11 cells in Twido, 256 x 30 cells in SoMachine Basic	Advisory	-	No
Unconditional action instruction between BLK and END_BLK	Error	-	No
OUT_BLK not followed by LD of a valid FB output or END_BLK	Error	-	No
FB cannot occupy first cell	-	-	Yes
FB on top of the rung, it replaces items occupying the cells	-	-	Yes
No logic above or below a FB	-	Error	No
XOR in first column	-	Error	No
Contacts and horizontal connectors in last column	-	Error	No
Down connectors in last row or last column	-	Error	No
Allow only valid subroutines 0 to 63	-	Error	No
Allow only valid labels 0 to 63	-	Error	No
Invalid operate expressions in operation block	-	Error	No
Invalid comparison expressions in comparison block	-	Error	No
Invalid address or symbol in contact and coil	-	Error	No
Invalid operand or expression with Ladder instruction	-	Error	No
Rung with no output action item	-	Error	No
Discontinuity between left and right power bars	-	Error	No
Dangling Ladder rung	-	Error	No

Situation	IL	Ladder	Rung reversible
Ladder rung contains items that are short-circuited using connectors	-	Error	No
All divergences that contain only boolean logic items must converge in reverse order	-	Error	No
FB has no input associated	-	Error	No
FB output pins cannot be connected together	-	Error	No
XOR connected to power bar	-	Error	No
Subroutine call and jump not the last output action item	Advisory	Error	No
Canonic rung that contains a FB with part of the FB in the last column	-	-	No
Canonic rung exceeds 7x11 cells in Twido, 256 x 30 cells in SoMachine Basic	Advisory	Error	No
OPEN and SHORT connected to the left node of subnetwork	-	Error	No
XOR connected to the left node of subnetwork	-	Error	No
There is not at least one existing LIST sentence that can represent the ladder rung	-	Error	No

How to Use the Source Code Examples

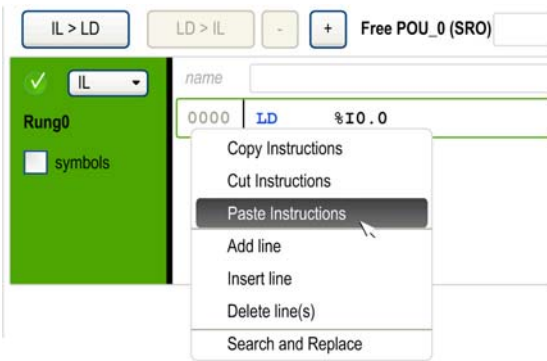

Overview

Except where explicitly mentioned, the source code examples contained in this book are valid for both the Ladder Diagram and Instruction List programming languages. A complete example may require more than one rung.

Reversibility Procedure

Only Instruction List source code is shown in this book.

To obtain the equivalent Ladder Diagram source code:

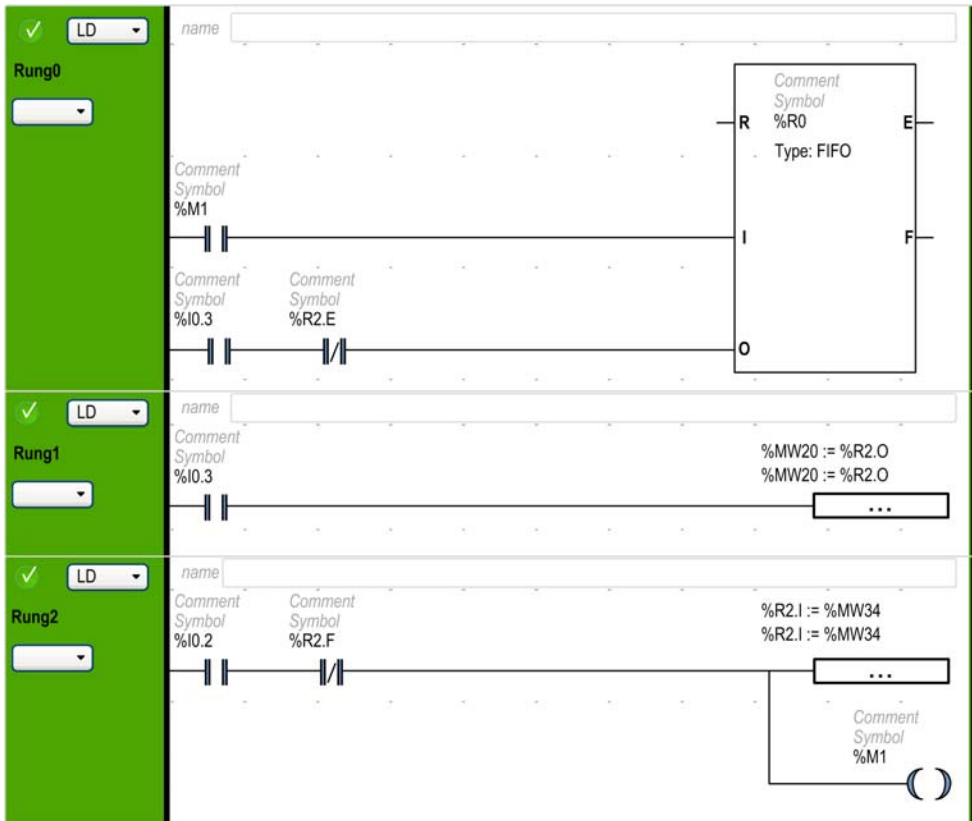
Step	Action
1	In SoMachine Basic, create a new POU containing an empty rung.
2	In this rung, click the LD > IL button to display Instruction List source code.
3	Select and copy (Ctrl+C) the source code for the first rung of the sample program.
4	Right-click on the line number 0000 of the first instruction and choose Paste Instructions to paste the source code into the rung:  <p>NOTE: Remember to delete the LD instruction from the last line of the rung if you have pasted the instructions by inserting the line(s) before the default LD operator.</p>
5	Click the IL > LD button to display the Ladder Diagram source code.
6	Repeat the previous steps for any additional rungs in the sample program. Click  on the toolbar to add new rungs.

Example

Instruction List program:

Rung	Source Code
0	BLK %R0 LD %M1 I LD %I0.3 ANDN %R2.E O END_BLK
1	LD %I0.3 [%MW20:=%R2.O]
2	LD %I0.2 ANDN %R2.F [%R2.I:=%MW34] ST %M1

Corresponding Ladder Diagram:



Section 6.3

Configuring Program Behavior and Tasks

What Is in This Section?

This section contains the following topics:

Topic	Page
Application Behavior	80
Tasks and Scan Modes	83

Application Behavior

Overview

You can configure the following aspects of how the application interacts with the logic controller:

- **Functional levels** (*see page 80*)
- **Startup** (*see page 81*)
- **Watchdog** (*see page 82*)
- **Fallback behavior** (*see page 82*)

Configuring Application Behavior

Follow these steps to configure the application behavior:

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	Select the Behavior item. Result: The Behavior properties appear in the lower central area of the Programming window.
3	Modify the properties as required.
4	Click Apply to save the changes.

Functional Levels

Your system could include logic controllers with different firmware versions, and therefore with different capability levels. SoMachine Basic supports functional level management to allow you to control the functional level of your application.

Select a level from the **Functional levels** list:

- **Level 3.1:** Contains enhancements (**Unconditional Start In Run** feature).
- **Level 3.0:** Contains enhancements (communications, modem, Remote Graphic Display) to the previous level of software and hardware.
- **Level 2.0:** Contains any enhancements and corrections over the previous level software and firmware. For example, for Pulse Train Output (PTO) support, it would be necessary to select this functional level or greater.
- **Level 1.0:** First release of the combination of the SoMachine Basic software and the compatible firmware version(s).

Startup

Specify how the program behaves following a restart of the logic controller:

- **Start In Previous State:** The logic controller starts in the state that it was in before it was stopped.
- **Start In Stop:** The logic controller does not automatically start application execution.
- **Start In Run (default):** The logic controller automatically starts application execution given run criteria, such as the presence and charge of a battery, are met.
- **Unconditional Start In Run:** The logic controller automatically starts application execution even if the controller battery is absent or discharged.

When using the Start In Run feature, the controller will start executing program logic when power is applied to the equipment. It is essential to know in advance how automatic reactivation of the outputs will affect the process or machine being controlled. Configure the Run/Stop input to help control the Start In Run feature. In addition, the Run/Stop input is designed to give local control over remote RUN commands. If the possibility of a remote RUN command after the controller had been stopped locally by SoMachine would have unintended consequences, you must configure and wire the Run/Stop input to help control this situation.

WARNING

UNINTENDED MACHINE START-UP

- Confirm that the automatic reactivation of the outputs does not produce unintended consequences before using the Start In Run feature.
- Use the Run/Stop input to help control the Start In Run feature and to help prevent the unintentional start-up from a remote location.
- Verify the state of security of your machine or process environment before applying power to the Run/Stop input or before issuing a Run command from a remote location.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

WARNING

UNINTENDED MACHINE OR PROCESS START-UP

- Verify the state of security of your machine or process environment before applying power to the Run/Stop input.
- Use the Run/Stop input to help prevent the unintentional start-up from a remote location.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

When using the Unconditional Start In Run feature, the controller will attempt to start executing program logic when power is applied to the equipment, independent of the reason the controller had previously stopped. This occurs even if there is no charge in the battery, or if the battery is not present. Therefore, the controller will start with all memory values re-initialized to zero or other predetermined default values. It is conceivable that if the controller attempts to restart, for example, after a short power outage, the values in memory at the time of the outage would be lost, and restarting the machine may have unintended consequences as there was no battery to maintain memory values. It is essential to know in advance how an unconditional start will affect the process or machine being controlled. Configure the Run/Stop input to help control the Unconditional Start In Run feature.

WARNING

UNINTENDED MACHINE OPERATION

- Conduct a thorough risk analysis to determine the effects, under all conditions, of configuring the controller with the Unconditional Start In Run feature.
- Use the Run/Stop input to help avoid an unwanted unconditional restart.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Watchdog

A watchdog is a special timer used to ensure that programs do not overrun their allocated scan time.

The watchdog timer has a default value of 250 ms. Specify the duration of the watchdog scan task. The possible range is 10...500 ms.

Fallback Behavior

Specify the fallback mode to use when the logic controller enters the STOPPED or an exception state for any reason.

Two fallback modes exist:

- By default, all outputs are set to the fallback values set in the configuration properties of embedded logic controller and expansion module outputs.
Refer to the *Programming Guide* of the logic controller or expansion module for information on configuring fallback values for outputs.
- Select **Maintain values** to keep each output in its current state when the logic controller enters the STOPPED or an exception state. In this mode, any fallback values configured for logic controller and expansion module outputs are ignored.

Tasks and Scan Modes

Overview

SoMachine Basic has the following scan modes:

- **Normal mode**
Continuous cyclic scanning mode (Freewheeling mode); a new scan starts immediately after the previous scan has completed.
- **Periodic mode**
Periodic cyclic scanning mode; a new scan starts only after the configured scan time of the previous scan has elapsed. Every scan is therefore the same duration.

SoMachine Basic offers the following task types:

- **Master task:** Main task of the application.
Master task is triggered by continuous cyclic scanning (in normal scan mode) or by the software times (in periodic scan mode) by specifying the scan period of 2...150 ms (default 100 ms).
- **Periodic task:** A short duration subroutine processed periodically.
Periodic tasks are triggered by software timers, so are configured by specifying the scan period of 5...255 ms (default 255 ms) in the periodic scan mode.
- **Event task:** A very short duration subroutine to reduce the response time of the application.
Event tasks are triggered by the physical inputs or the HSC function blocks. These events are associated with embedded digital inputs (%I0.2...%I0.5) (rising, falling or both edges) or with the high speed counters (%HSC0 and %HSC1) (when the count reaches the high speed counter threshold). You can configure 2 events for each HSC function block.

Periodic tasks and events are configured in periodic scan mode. Master task can be configured in either normal scan mode or periodic scan mode.

Tasks Priorities

This table summarizes the task types and their priorities:

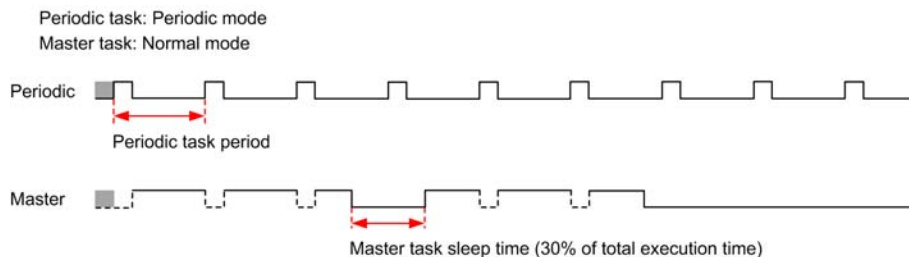
Task Type	Scan Mode	Triggering Condition	Configurable Range	Maximum Number of Tasks	Priority
Master	Normal	Normal	Not applicable	1	Lowest
	Periodic	Software timer	2...150 ms		
Periodic	Periodic	Software timer	5...255 ms	1	Higher than master task and lower than event tasks
Event	Periodic	Physical inputs	%I0.2...%I0.5	4	Highest
		%HSC function blocks	2 events per %HSC object	4	

Events Priorities

Refer to Event Priorities and Queues ([see page 106](#)).

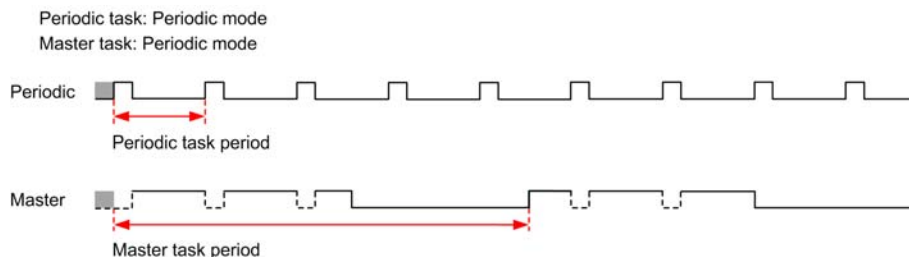
Master Task in Normal Scan Mode

This graphic shows the relationship between master tasks and periodic task execution when the master task is configured in normal scan mode:



Master Task in Periodic Scan Mode

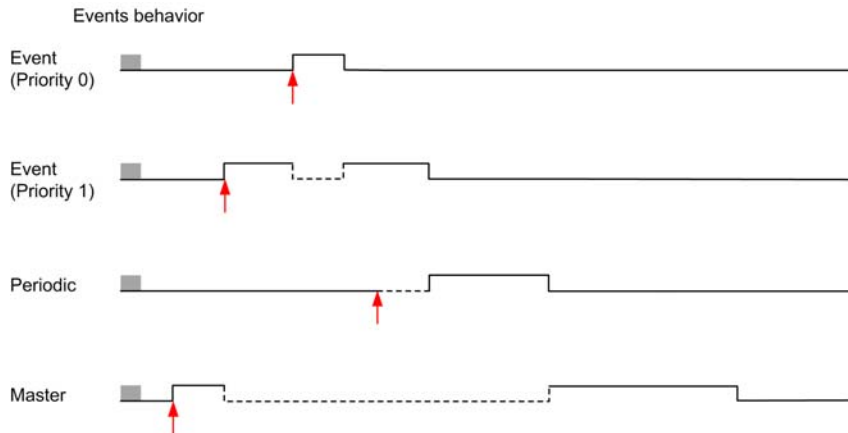
This graphic shows the relationship between master tasks and periodic tasks when the master task is configured in periodic scan mode:



Event Priority Over Master and Periodic Tasks

Event priorities control the relationship between the event tasks, master tasks, and periodic tasks. The event task interrupts the master task and periodic task execution.

This figure shows the relationship between event tasks, master tasks, and periodic tasks in the periodic mode:



The event tasks are triggered by a hardware interruption that sends a task event to the event task.

Section 6.4

Managing POUs

What Is in This Section?

This section contains the following topics:

Topic	Page
POUs	87
Managing POUs with Tasks	88
Managing Rungs	90
Free POUs	93

POUs

Overview

A Program Organization Unit (POU) is a reusable object used in a program. Each POU consists of a variable declaration and a set of instructions in the source code of a supported programming language.

One POU always exists and is linked to the master task of the program. This POU is then called automatically whenever the program starts.


You can create additional POUs containing other objects, for example, functions or function blocks.

When first created, a POU can be either:

- associated with a task ([see page 88](#)), or
- a Free POU ([see page 93](#)). A Free POU is not associated with a specific task or event. A Free POU can, for example, contain library functions that are maintained independently of the main program. Free POUs are called from within programs as either subroutines or jumps. A periodic task ([see page 100](#)) is a subroutine that is implemented as a Free POU.

Managing POUs with Tasks

Creating a New POU Associated with a Task

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	<p>Add a new POU by one of the following methods:</p> <ul style="list-style-type: none"> ● Right-click on the Master Task and choose Add POU from the contextual menu that appears. ● Select the Master Task and click  (Add POU button) on the toolbar at the top of the Tasks tab. <p>Result: A new POU is added to the program structure immediately below the default/last POU in the Master Task. The default name is <i>n</i> - New POU, where <i>n</i> is an integer incremented each time a POU is created.</p>
3	If required to reposition a POU in the Master Task , select a POU and click the UP or DOWN button on the toolbar at the top of the Tasks tab to move the selected POU up or down in the program structure.

Copying and Pasting an Existing POU Associated with a Task

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	Right-click on an existing POU in the Master Task and choose Copy POU from the contextual menu that appears.
3	<p>Right-click on the Master Task and choose Paste POU from the contextual menu that appears.</p> <p>Result: A new POU is added to the program structure immediately below the default/last POU in the Master Task with the same name as the copied POU.</p>

Renaming a POU



Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	<p>Edit the POU name by one of the following methods:</p> <ul style="list-style-type: none"> ● Right-click on a POU and choose Rename POU from the contextual menu that appears. ● Double-click a POU. ● Select a POU and double-click the POU name in the programming workspace.
3	Type the new name for the POU and press ENTER.

Removing a POU


Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	Right-click on a POU in the Master Task and choose Delete POU from the contextual menu that appears.
3	Click Yes to confirm deletion.

Managing Rungs

Creating a Rung

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	<p>Add a rung in a POU by any of the following methods:</p> <ul style="list-style-type: none"> ● Right-click on a POU and choose Add Rung from the contextual menu that appears. ● Select a POU and click  (Add Rung button) on the toolbar at the top of the Tasks tab. ● Select a POU and click  (Create a new Rung button) on the toolbar at the top of the programming workspace. <p>Result: A new rung is added to the program structure immediately below the last rung.</p>
3	If required to reposition a rung in a POU, select a rung and click the UP or DOWN button on the toolbar at the top of the Tasks tab to move the selected rung up or down in the program structure.
4	The rung is given sequence identifier, such as Rung0. You may additionally add a rung comment to identify the rung by clicking the rung header.
5	The default programming language is LD (ladder). To select a different programming language for this rung, click LD and choose a different programming language.
6	<p>If this rung is to be called with a JUMP instruction, assign a label to the rung by clicking the drop-down button below the rung sequence identifier Rungx, where <i>x</i> is the rung number in a POU, and choose %L from the list.</p> <p>Result: The rung is labeled as %Ly, where <i>y</i> is the label number. %L appears on the button and the label number <i>y</i> appears in suffix with the button.</p> <p>NOTE: The label number is incremented by 1 as you define the next label.</p> <p>To modify the label number, double-click the label number in a rung and enter the new number and then press ENTER.</p>

Inserting a Rung Above an Existing Rung

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	Select an existing rung in the Programming workspace.
3	<p>Click  (Insert a new Rung button) on the toolbar at the top of the programming workspace.</p> <p>Result: A new rung appears above the selected rung.</p>

Step	Action
4	The rung is given sequence identifier, such as <code>Rung0</code> . You may additionally add a rung comment to identify the rung by clicking the rung header.
5	The default programming language is LD (ladder). To select a different programming language for this rung, click LD and choose a different language.
6	<p>If this rung is to be called with a <code>JUMP</code> instruction, assign a label to the rung by clicking the drop-down button below the rung sequence identifier Rungx, where <i>x</i> is the rung number in a POU, and choose %L from the list.</p> <p>Result: The rung is labeled as %Ly, where <i>y</i> is the label number. %L appears on the button and the label number <i>y</i> appears in suffix with the button.</p> <p>NOTE: The label number is incremented by 1 as you define the next label.</p> <p>To modify the label number, double-click the label number in a rung and enter the new number and then press ENTER.</p>

Copying a Rung

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	Right-click on the rung to copy and choose Copy selected rung from the contextual menu that appears.
3	<p>Right-click on a POU and choose Paste Rung from the contextual menu that appears.</p> <p>Result: A copy of the rung is inserted with no label.</p> <p>NOTE: Label of the rung is not copied when you copy a rung.</p>



NOTE: You can also copy and paste rungs in the **Programming** window:

Step	Action
1	Right-click on the rung to copy and choose Copy selected rung .
2	Right-click anywhere in the programming workspace and choose Paste Rung .

Renaming a Rung

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	<p>Edit the rung name by one of the following methods:</p> <ul style="list-style-type: none"> ● Right-click on a rung and choose Rename Rung from the contextual menu that appears. ● Double-click a rung. ● Select a rung and double-click the rung name or the text name in the programming workspace.
3	Type the new name for the rung and press ENTER.

Deleting a Rung

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	<p>Delete a rung by one of the following methods:</p> <ul style="list-style-type: none"> ● Right-click on a rung and choose Delete Rung from the contextual menu that appears. ● Select a rung and click  (Delete Rung button) on the toolbar at the top of the Tasks tab. ● Select a rung and click  (Delete the Rung button) on the toolbar at the top of the programming workspace. ● Right-click on a rung in the programming workspace and choose Delete the selected rung from the contextual menu that appears.
3	If the rung is not empty, you are prompted to confirm deleting the rung. Click Yes to confirm deletion, or click No to cancel the operation.

Free POU's

Introduction

In SoMachine Basic, a Free POU is a special type of POU that is not explicitly associated with a task:



Each Free POU is implemented as a subroutine made up of 1 or more rungs written in any of the programming languages supported by SoMachine Basic.

Free POU's are consumed when:

- Called using a subroutine call (SRi) from within a program rung
- Configured as the periodic task
- Configured as an event task, for example, the subroutine for threshold 0 of a High Speed Counter (HSC) function block (%HSCi.TH0)

When consumed as periodic or event tasks, the Free POU subroutine is automatically moved from the **Free POU's** area of the **Tasks** window to the **Periodic Task** or **Events** area of the window, respectively.

When no longer consumed as a periodic task or event task, the subroutine moves back to the **Free POU's** area and available to be consumed by other tasks or events.

Creating a New Free POU

Proceed as follows to create a new Free POU:

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	Right-click on Free POU's and choose Add Free POU from the contextual menu that appears. Result: A new POU with the default name "Free POU_0" and default subroutine number "SR0" appears below the Free POU's branch and a new rung appears in the Programming workspace.
3	Optionally, right-click on the new POU and choose Rename POU , then type a new name for the POU and press Enter. The name of the Free POU is also updated in the rung that appears in the Programming workspace.
4	Optionally, type a comment (<i>see page 146</i>) to associate with the Free POU.

Step	Action
5	Select Subroutine number to the right of the comment box and choose a subroutine number from the list. Result: The POU description in the Free POUs list is updated with the subroutine number chosen, for example "SR11".
6	Create the rungs and source code for the Free POU in the programming language of your choice.

Copying and Pasting an Existing POU

Proceed as follows to copy and paste an existing POU associated with a task to create a Free POU:

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	Right-click on an existing POU and choose Copy POU .
3	Right-click on Free POUs and choose Paste POU . Result: A new Free POU with the name "Free POU_x", where x is the next available Free POU number, and default subroutine number "SRx", where x is the next available subroutine number, appears below Free POUs . All rungs of the POU are automatically associated with the new Free POU subroutine number.

Assigning Free POUs to Events or Periodic Tasks

By default, Free POUs and subroutines are not associated with any events or tasks.

Refer to [Creating Periodic Task \(see page 100\)](#) for information on how to associate a Free POU with a periodic task.

Refer to [Creating Event Task \(see page 107\)](#) for information on how to associate a Free POU with an event.

Section 6.5

Master Task

What Is in This Section?

This section contains the following topics:

Topic	Page
Master Task Description	96
Configuring Master Task	97

Master Task Description

Overview

The master task represents the main task of the application program. It is obligatory and is created by default. The master task is made up of sections and subroutines represented within Program Organizational Units (POUs). Each POU of the master task can be programmed in any of the supported programming languages.

Procedure

For	Refer To
Creating a new POU in the master task	Creating a New POU Associated with a Task (see page 88)
Renaming a POU in the master task	Renaming a POU (see page 88)
Removing a POU from the master task	Removing a POU (see page 89)

Configuring Master Task

Procedure

Follow these steps to configure the master task:

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	Select the Master Task item. Result: The Master Task properties appear in the lower central area of the SoMachine Basic window.
3	Modify the properties as required.
4	Click Apply to save the changes.

Master Task Properties

Scan Mode

Choose the scan mode to use for the program:

- **Normal:** When a logic controller is in normal (freewheeling) scan mode, a new scan starts immediately after the previous scan has completed.
- **Periodic:** In periodic scan mode, the logic controller waits until the configured scan time has elapsed before starting a new scan. Every scan is therefore the same duration. Specify the scan **Period** for the periodic scan mode of 2...150 ms. The default value is 100 ms.

The default scan mode is Normal.

System Bits and Words Controlling the Master Task

The master task can be controlled by system bits (%S) and system words (%SW):

This table lists the system bits:

System Bits	Description
%S11	Watchdog overflow
%S19	Scan period overrun (periodic scan mode)

This table lists the system words:

System Words	Description
%SW0	Logic controller scan period (periodic scan mode)
%SW27	Time spent (in ms) in the system during the last master task scan cycle
%SW30	Last scan time. Show the execution time (in ms) of the last controller scan cycle, that is, the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a master task scan cycle

System Words	Description
%SW31	Maximum scan time. Show the execution time (in ms) of the longest controller scan time since the last cold start of the logic controller.
%SW32	Minimum scan time. The execution time (in ms) of the shortest controller scan time since the last cold restart of the logic controller.

Refer to the *Programming Guide* for your hardware platform for a complete list of system bits and words and their meaning.

Section 6.6

Periodic Task

What Is in This Section?

This section contains the following topics:

Topic	Page
Creating Periodic Task	100
Configuring Periodic Task Scan Duration	102


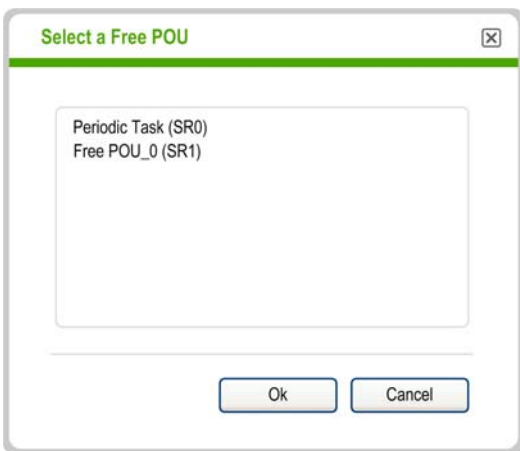
Creating Periodic Task

Overview

A periodic task is a subroutine, usually of short duration, that is processed periodically by the master task. In SoMachine Basic, this subroutine is implemented as a Free POU (see page 93). The subroutine can be written in any of the programming languages supported by SoMachine Basic.


NOTE: The maximum number of periodic tasks that can exist in a program is determined by the logic controller hardware. Refer to the *Programming Guide* of the logic controller for details.

Assigning a Subroutine to a Periodic Task

Step	Action
1	Create a new Free POU (see page 93) containing the periodic task subroutine.
2	Select the Tasks tab in the left-hand area of the Programming window.
3	<p>Assign a subroutine to the periodic task by one of the following methods:</p> <ul style="list-style-type: none"> ● Select the Periodic Task and click  (Assign Free POU button) on the toolbar at the top of the Tasks tab. ● Right-click the Periodic Task and choose Assign Free POU from the contextual menu that appears. <p>Result: The Select a Free POU window is displayed:</p>  <p>NOTE: You can directly add a Free POU to the periodic task. Right-click the Periodic Task and choose Add Free POU from the contextual menu that appears. In this case, a Free POU is created and assigned to the periodic task.</p>

Step	Action
4	<p>Select a Free POU to assign to the periodic task and click OK.</p> <p>Result: The selected subroutine is assigned to the Periodic Task and no longer available in the Free POU branch of the Tasks tab.</p> <p>For example, if the Free POU "Free POU_0" containing the subroutine SR4 is assigned to the periodic task, the Free POU_0 (%SR4) subroutine moves from the Free POU branch to the Periodic Task branch of the Tasks tab.</p>

Removing a Subroutine from a Periodic Task

Step	Action
1	Click the Tasks tab in the left-hand area of the Programming window.
2	<p>Remove the subroutine from the Periodic Task by one of the following methods:</p> <ul style="list-style-type: none"> • Select the Periodic Task and click  (Unassign Free POU button) on the toolbar at the top of the Tasks tab. • Right-click the Periodic Task and choose Unassign Free POU from the contextual menu that appears. <p>Result: The selected subroutine is removed from the Periodic Task and available as a Free POU in the Free POUs branch of the Tasks tab.</p>

Configuring Periodic Task Scan Duration

Procedure

Follow these steps to configure scan duration for periodic task:

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	Select the Periodic Task item. Result: The Periodic Task properties appear in the lower central area of the SoMachine Basic window.
3	Modify the properties as required.
4	Click Apply to save the changes.

Periodic Task Properties

Specify the scan **Period** for the periodic task from 5...255 ms. The default value is 255 ms.

Section 6.7

Event Task

What Is in This Section?

This section contains the following topics:

Topic	Page
Overview of Event Tasks	104
Event Sources	105
Event Priorities and Queues	106
Creating Event Task	107

Overview of Event Tasks

Introduction

An event task:

- Is a part of a program executed when a given condition is met (event source)
- Has a higher priority than the main program
- Produces a rapid response time, enabling the overall response time of the system to be reduced.

Description of an Event

An event is composed of:

- An *event source*: a software or hardware condition that interrupts the program when the event is triggered
- A *POU*: an independent program entity (subroutine) associated with an event
- An *event queue*: used to store a list of events until they are executed
- A *priority level*: a priority assigned to events to determine the order in which they are executed.

Event Sources

Overview

9 event sources are available:

- 4 linked to selected physical inputs of the logic controller
- 4 linked to the HSC function block thresholds (2 events per %HSC instance)
- 1 periodic condition.

An event source is always attached to a single event. When an event is triggered it is immediately detected by the controller, which then executes the subroutine associated with the event.

Physical Input Events of a Logic Controller

The embedded digital inputs %I0.2, %I0.3, %I0.4 and %I0.5 of a logic controller can be configured as event sources.

These event sources can be configured to:

- Trigger events on detection of a rising edge, falling edge, or both rising and falling edges
- Assign a priority to the event
- Identify the subroutine associated with the event.

For further details on configuring input events, refer to the *Programming Guide* of the logic controller.

Output Event of an %HSC Function Block

The threshold outputs TH0 and TH1 of the %HSC function block can be used as event sources. Outputs TH0 and TH1 are respectively set to:

- 1 when the value is greater than threshold S0 and threshold S1,
- 0 when the value is less than threshold S0 and threshold S1.

A rising or falling edge of these outputs can activate an event process.

For further details on configuring output event, refer to the *Programming Guide* of the logic controller.

Periodic Event

This event source periodically executes a designated program section (subroutine). This subroutine has a higher priority than the master task. However, the periodic event source has lower priority than all the other event sources.

For further details on configuring this event, refer to Periodic Task ([see page 99](#)).

Event Priorities and Queues


Event Priorities and Queues

Events have one of 8 possible priorities, from 7 (the lowest) to 0 (the highest).

Assign a priority to each event source. Only one event source at any one time can have priority 0. The other events therefore have lower priority, and their order of execution depends on their relative priorities and the order in which they are detected.

The priority of an event controls the relationship between event task execution. All event tasks interrupt both master and periodic task execution. For more information, refer to Event Priority Over Master and Periodic Tasks ([see page 85](#)).

NOTE: Care must be exercised when writing to global areas of memory or affecting I/O values when event tasks are called during the execution of other tasks. Modifying values that are otherwise used in the other tasks could affect logical outcomes of those tasks adversely.

 WARNING
<p>UNINTENDED EQUIPMENT OPERATION</p> <p>Thoroughly test and validate all tasks (Master, Periodic and any Event tasks) and the interactive affect they have on one another before putting your application into service.</p> <p>Failure to follow these instructions can result in death, serious injury, or equipment damage.</p>

For configuring the event task priorities, refer to *Programming Guide* of your controller.

Event Queue Management

Each time an interrupt linked to an event source appears, the following sequence is launched:

Step	Description
1	Interrupt management: <ul style="list-style-type: none"> ● recognition of the physical interrupt, ● store the event in the suitable event queue, ● verification that no event of the same priority is pending (if so, the event stays pending in the queue).
2	Save the context.
3	Execution of the programming section (subroutine labeled SRi:) linked to the event.
4	Update the outputs
5	Restore the context

Before the context is re-established, all the events in the queue must be executed.

Creating Event Task

Overview

You can view currently configured event sources, subroutines currently attached to events, and check the current status of events using system bits and words.


To view the currently assigned event sources and subroutines (Free POU's) assigned to events, do the following action:

Step	Action
1	Select the Tasks tab in the left-hand area of the Programming window.
2	<p>Select Events:</p> <pre> [-] Events %HSC0.TH0 : %HSC0.TH1 : %I0.2 : [-] %I0.3 : Free POU_0 Rung0 </pre> <p>NOTE: Configured event sources that have not yet been assigned a subroutine appear in red.</p>

NOTE: Only embedded controller inputs/outputs can be used in an event subroutine.


Assigning a Free POU to an Event Source

Proceed as follows to assign a Free POU to a configured event source:

Step	Action
1	Create a new Free POU (<i>see page 93</i>) containing the subroutine to use for the event.
2	Select the Tasks tab in the left-hand area of the Programming window.
3	<p>Assign a subroutine to the event source by one of the following methods:</p> <ul style="list-style-type: none"> ● Select the event source in the Events list and click  (Assign Free POU button) on the toolbar at the top of the Tasks tab. ● Right-click the event source in the Events list and choose Assign Free POU from the contextual menu that appears. <p>Result: The Select a Free POU window is displayed:</p> <div data-bbox="323 578 847 1023" style="border: 1px solid gray; padding: 10px; margin: 10px 0;"> <p style="text-align: center; margin: 0;">Select a Free POU ✕</p> <hr style="border: 2px solid green; margin: 5px 0;"/> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p>Event_TH0 (SR0)</p> <p>Free POU_0 (SR1)</p> </div> <div style="text-align: center; margin-top: 20px;"> <input type="button" value="Ok"/> <input type="button" value="Cancel"/> </div> </div> <p>NOTE: You can directly add a Free POU to the event source. Right-click the event source in the Events list and choose Add Free POU from the contextual menu that appears. In this case, a Free POU is created and assigned to the event source.</p>
4	<p>Select a Free POU to assign to the event source and click OK.</p> <p>Result: The selected subroutine is assigned to the event source and no longer available in the Free POU branch of the Tasks tab.</p> <p>For example, if the Free POU “Free POU_0” containing the subroutine SR1 is assigned to the event source, the Free POU_0 (%SR1) subroutine moves from the Free POU branch to the event source branch of the Tasks tab.</p>

Removing a Subroutine from an Event

To remove the association between a subroutine and an event source, follow these steps:

Step	Action
1	Click the Tasks tab in the left-hand area of the Programming window.
2	<p>Remove the subroutine from the event source by one of the following methods:</p> <ul style="list-style-type: none"> • Select the event source in the Events list and click  (Unassign Free POU button) on the toolbar at the top of the Tasks tab. • Right-click the event source in the Events list and choose Unassign Free POU from the contextual menu that appears. <p>Result: The selected subroutine is removed from the event source and available as a Free POU in the Free POUs branch of the Tasks tab.</p>

Checking Events with System Bits and Words

The following system bits are used to check the events:

System Bit	Description
%S31	Used to execute or delay an event
%S38	Used to decide whether to place events in the event queue
%S39	Used to determine if events are lost

The following system words are used to check the events:

System Word	Description
%SW48	The number of events that have been executed since the last cold start of the logic controller (counts all events except periodic events).

The values of %S39 and %SW48 are reset to 0 and the values of system bits %S31 and %S38 are set to their initial state 1 following a cold restart or after an application is loaded. Their values remain unchanged after a warm restart. In all cases, the event queue is reset.

Section 6.8

Using Tools

What Is in This Section?

This section contains the following topics:

Topic	Page
Program Messages	111
Animation Tables	113
Memory Objects	116
System Objects	118
I/O Objects	119
Software Objects	120
PTO Objects	121
Communication Objects	122
Search and Replace	123
Symbol List	125
Memory Consumption View	129
Rung Templates	131

Program Messages

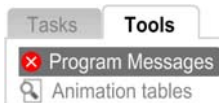
Overview

SoMachine Basic continuously compiles the source code displayed in the **Programming** tab into a program ready to be downloaded to the logic controller.

If errors or advisories are detected, an icon is displayed in the **Programming** tab:



The icon is also displayed in the **Tools** tab next to **Program Messages**:



The icon that is displayed depends on the message severity:

Icon	Meaning
	Advisory
	Error

If both advisory and error messages are detected, only the Error icon is displayed.

Displaying Program Messages

To display a list of error and advisory messages:

Step	Action
1	Click the icon on the Programming tab or: Click Tools → Program Messages A list of messages is displayed in the lower central area of the Programming window..
2	In the Messages area, click the Advisory button to display advisory messages, or the Error button to display error messages. Click the button again to hide the list of messages.

Rung Compilation Status

SoMachine Basic also displays the compilation status of each rung in the program individually.

If a rung compiles successfully and there are no messages to display, a green tick symbol appears:

<input checked="" type="checkbox"/> IL Rung0 <input type="checkbox"/> symbols	name			
	0000	LD	%I0.0	Comment
	0001	ST	%Q0.0	Comment

An Advisory icon appears if SoMachine Basic cannot compile the program because the rung is incomplete, for example, it does not contain a final instruction such as an END, CALL, or Jump:

<input type="checkbox"/> IL Rung0 <input type="checkbox"/> symbols	name			
	0000	LD	%I0.0	Comment
	0001	ANDN	%I0.1	Comment

An Error icon appears if SoMachine Basic detects errors that prevent successful compilation of the rung:

<input checked="" type="checkbox"/> IL Rung0 <input type="checkbox"/> symbols	name			
	0000	BLK	%SBR0	Comment
	0001	LD	%S6	Comment
	0002	CU		Comment

Advisory and Error icons are also displayed next to the name of each rung with errors in the **Tasks** tab:

- 1 - M_ZeroPressureAccumulator
 - Rung0
 - Rung1
 - Rung2
 - Rung3 - Rung_1
 - Rung4 - Rung_3
 - Rung5
 - Rung6 - Rung_2
 - Rung7
 - Rung8

Animation Tables

Overview

You can manually add objects to animation tables. An animation table allows you to:

- select objects to be displayed in the Trace window ([see page 170](#)).
- view symbols associated with objects.
- view and modify the real-time values of certain object types when SoMachine Basic is connected to the logic controller and the program is running (online mode).

Animation tables are a component of a SoMachine Basic application, and so are uploaded to the logic controller together with the program. This allows the objects and values stored in animation tables to be retrieved when an application is later downloaded from the logic controller.

Animation table								
%I0.0		Add		Insert		Time Base	5	Open Trace window
Used	Trace	Address	Symbol	Value	Force	Comment		
<input checked="" type="checkbox"/>	<input type="checkbox"/>	%MW50	ADDRESS_MEM	0				
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	%MW610	CONTROL_CMD	0				
<input checked="" type="checkbox"/>	<input type="checkbox"/>	%M16	MODBUS_READ	0				
<input checked="" type="checkbox"/>	<input type="checkbox"/>	%MW61	SPEED_VALUE	0				
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	%MW40	CMD	0		Control World		

Creating an Animation Table

Step	Action
1	Select the Tools tab in the left-hand area of the Programming window.
2	Right-click on Animation tables and choose Add new animation table from the contextual menu that appears. Result: A new animation table item appears below the Animation Tables area of the Tools window, and a properties window appears in the lower central area of the window.

Adding Items To an Animation Table

Step	Action
1	Select the Tools tab in the left-hand area of the Programming window.
2	Select the animation table to configure in the Animation tables area of the Tools window. Result: The properties window appears in the lower central area of the window.
3	To add a new item to the bottom of the animation table, type the object name into the text box and press Enter, or click Add . The following objects can be added to an animation table: <ul style="list-style-type: none"> ● I/O objects ● Bit strings (example: %Mx:L where L is the bit count, multiple of 8) ● Word tables (example: %MWx:L where L is the word count) ● Bits of words (example: %MWx:X where X is the offset of the bit)
4	To add a new object immediately above an existing object, select a row in the animation table, type the name of the object to add into the text box, and click Insert .

Animation Table Properties

This table describes the properties of the animation tables:

Parameter	Editable	Value	Description
Used	No	True/False	Indicates whether the object is currently being used in a program.
Trace	Yes ⁽¹⁾	True/False	Select the object to trace in the Trace window (<i>see page 170</i>).
Address	No	Object address	Displays the address of the object.
Symbol	No	A valid symbol	The name of the symbol associated with this object, if defined.
Value	Yes ⁽²⁾	Current value	The current value of the object. If the object type has read/write access and you are in online mode (<i>see page 27</i>), double-click and type a new object value if required. The value of the object is updated in real time in the program running in the logic controller. See <i>Modifying Real-Time Values (see page 172)</i> for details.
Force	Yes ⁽²⁾	Force to 0 Force to 1 Not Forced	Only appears for digital inputs and digital outputs. Only editable when in online mode (<i>see page 27</i>). Allows you to force the value of the input or output to 0 or 1 as required. Choose Not Forced to remove any forcing currently applied to the address.
Comment	No	A valid comment	The comment associated with this object, if defined.
<p>(1) You can select up to 8 objects.</p> <p>(2) Depending on the object type and on whether you are in online mode.</p>			

Configuring Items In an Animation Table

To search for and optionally replace an object in an animation table, right-click on the object and choose **Search and Replace**. Refer to Search and Replace ([see page 123](#)) for further details.

To remove an object from an animation table, right-click on the object and choose **Remove from animation table**.

Renaming an Animation Table

Step	Action
1	Right-click on the animation table to rename in the Animation tables area of the Tools window and click Rename animation table .
2	Type the new name of the animation table and press Enter.

Deleting an Animation Table

Step	Action
1	Right-click on the animation table to delete in the Animation tables area of the Tools window and click Delete animation table .
2	Click Yes to confirm.

Opening the Trace Window

Step	Action
1	Select up to 8 objects in the Trace column of an animation table.
2	Connect (see page 181) to the logic controller or launch the simulator (see page 186).
3	Select a value in the Time Base list. This determines the refresh frequency of the Trace window (see page 170), in seconds.
4	Click Open Trace window . The Trace window is displayed.

Memory Objects

Overview

Memory objects include:

- Memory bits
- Memory words
- Constant words

Selecting the Memory Allocation Mode

Before viewing or updating the properties of memory objects, choose the memory allocation mode (*see page 70*) to use.

Memory Bit Properties

This table describes each parameter of the **Memory bits** screen:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the memory bit is currently being used in a program.
Address	No	Refer to Bit Objects	N/A	Displays the address of the memory bit, where <i>x</i> is the number of memory bits supported by the logic controller.
Symbol	Yes	A valid symbol	<i>None</i>	Allows you to associate a symbol with this memory bit.
Value	Yes	Refer to Bit Objects.	0	The value of this memory bit.
Comment	Yes	A valid comment	None	Allows you to associate a comment with this memory bit

Memory Word Properties



First choose the memory word type to display properties for:

- **%MW**. Memory words
- **%MD**. Double words
- **%MF**. Floating-point words

This table describes the properties of **Memory words**:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the memory word is currently being used in a program.
Equ Used	No	True/False	False	Indicates whether the memory word is currently being used in an equation.
Address	No	Refer to Word Objects	N/A	Displays the address of the memory word.
Symbol	Yes	A valid symbol	None	Allows you to associate a symbol with this memory word.
Value	Yes	Refer to Word Objects.	0	The value of this memory word.
Comment	Yes	A valid comment	None	Allows you to associate a comment with this memory word.

Constant Word Properties



First choose the constant word type to display properties for:

- **%KW**. Constant words.
- **%KD**. Double constant words
- **%KF**. Floating-point constant words.

This table describes each parameter of the **Constant words** screen:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the constant word is currently being used in a program.
Equ Used	No	True/False	False	Indicates whether the constant word is currently being used in an equation.
Address	No	Refer to Word Objects	N/A	Displays the address of the constant word.
Symbol	Yes	A valid symbol	None	Allows you to associate a symbol with this constant word.
Value	Yes	Refer to Word Objects	0	The value of this constant word.
Comment	Yes	A valid comment	None	Allows you to associate a comment with this constant word.

System Objects

Overview

System bits and words are specific to the logic controller. For details, refer to the *Programming Guide* of your logic controller.

I/O Objects

Overview

The following object types are hardware-specific and depend on the logic controller being used:

- Digital inputs and outputs
- Analog inputs and outputs
- Advanced function blocks such as fast counters, high-speed counters, and pulse generators.

For more details, refer to the *Programming Guide* and *Advanced Functions Library Guide* of your logic controller.

Software Objects

Overview

SoMachine Basic supports the following generic software objects:

Object	Description
Timers	Used to specify a period of time before doing something, for example, triggering an event.
Counters	Provides up and down counting of events.
Messages	Allows communication with external devices.
LIFO/FIFO Registers	A memory block that can store up to 16 words of 16 bits each in FIFO or LIFO modes.
Drum Registers	Operates on a principle similar to an electromechanical Drum Controller, which changes step according to external events. On each step, the high point of a cam gives a command which is executed by the logic controller.
Shift Bit Registers	Provides a left or right shift of binary data bits (0 or 1).
Step Counters	Provides a series of steps to which actions can be assigned.
Schedule Blocks	Used to control actions at a predefined month, day, and time.
PID	Allows regulation of the Proportional Integral Derivative (PID) function.

These function blocks are described in the *SoMachine Basic Generic Functions Library Guide*.

Selecting the Memory Allocation Mode

Before viewing or updating the properties of software objects, choose the memory allocation mode ([see page 70](#)) to use.

PTO Objects

Overview

The PTO objects provide the function blocks used for programming the PTO functions. The PTO function blocks are categorized as:

- Motion
These function blocks control motions of the axis. For example, power to axis, movement of the axis, and so on.
- Administrative
These function blocks control the status and diagnostics of the axis movement. For example, status and value of actual velocity, actual position, axis control detected errors, and so on.

For more details on the PTO function blocks, refer to the *Advanced Function Library Guide* of your controller.

Communication Objects

Overview

Communication objects are used to communicate with Modbus devices, and to send/receive messages in character mode (ASCII).

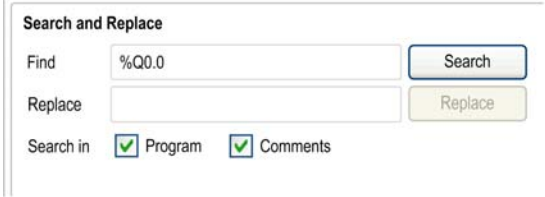
For details, refer to the chapter Communication Objects (see *SoMachine Basic, Generic Functions Library Guide*).

Search and Replace

Overview

Search and Replace allows you to find all occurrences of an object used anywhere in a program and optionally replace it with a different object.

Searching and Replacing Items

Step	Action
1	Select the Tools tab in the left-hand area of the Programming window. It is also possible to invoke the search and replace function from various other locations in SoMachine Basic, for example, by right-clicking on an entry in an animation table (<i>see page 113</i>) and selecting Search & Replace .
2	<p>You can use any of the following methods to display the Search and Replace window:</p> <ul style="list-style-type: none"> • Click Search and Replace in the Tools tab of the Programming window. • Right-click on a rung or a selected item in the rung and click Search and Replace in the context menu that appears. • Right-click on a line in the properties window of any object and click Search and Replace in the context menu that appears. <p>This graphic shows the Search and Replace window:</p> 
3	<p>In the Find box type the object or symbol name to find. The Find field is pre-filled if the search was started by right-clicking on a selected item in a rung or an item in a properties window of an object.)</p> <p>You can use the following wildcard characters:</p> <ul style="list-style-type: none"> • Asterisk (*). Replaces 0 or more characters in the search term. For example, %MW1* would find both %MW1 and %MW101. • Question mark (?). Replaces exactly 1 character in the search term. For example, typing COIL?2 would find COIL12 but not COIL012
4	Optionally, in the Replace box type a replacement object or symbol name.
5	Select Program to search for the item within the source code of the current program. Select Comments to search for the item within program comments.

Step	Action									
6	<p>Click Search or Replace. You can also press ENTER to start the search. Replace button is enabled only when the replacement object or symbol name is given in the Replace box.</p> <p>All items found are listed in the Results list:</p> <p style="text-align: right;"><input type="checkbox"/> Show symbols</p> <table border="1" data-bbox="322 342 1156 451"> <thead> <tr> <th>POU</th> <th>Rung</th> <th>Code</th> </tr> </thead> <tbody> <tr> <td>POU_0</td> <td>Rung_0</td> <td>%Q0.0</td> </tr> <tr> <td>POU_0</td> <td>Rung_1</td> <td>LD %Q0.0</td> </tr> </tbody> </table>	POU	Rung	Code	POU_0	Rung_0	%Q0.0	POU_0	Rung_1	LD %Q0.0
POU	Rung	Code								
POU_0	Rung_0	%Q0.0								
POU_0	Rung_1	LD %Q0.0								
7	<p>Optionally, select Show symbols to display instead any symbols defined for objects:</p> <p style="text-align: right;"><input checked="" type="checkbox"/> Show symbols</p> <table border="1" data-bbox="322 561 1156 670"> <thead> <tr> <th>POU</th> <th>Rung</th> <th>Code</th> </tr> </thead> <tbody> <tr> <td>POU_0</td> <td>Rung_0</td> <td>OUTPUT</td> </tr> <tr> <td>POU_0</td> <td>Rung_1</td> <td>LD OUTPUT</td> </tr> </tbody> </table>	POU	Rung	Code	POU_0	Rung_0	OUTPUT	POU_0	Rung_1	LD OUTPUT
POU	Rung	Code								
POU_0	Rung_0	OUTPUT								
POU_0	Rung_1	LD OUTPUT								
8	<p>Click on any of the listed results to jump directly to the line of code in the program.</p>									

Symbol List

Overview

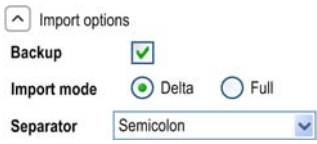
You can display a list of all the symbols that have been associated with objects in your program. All objects with symbols are displayed, except the system bits (%S) and system words (%SW).

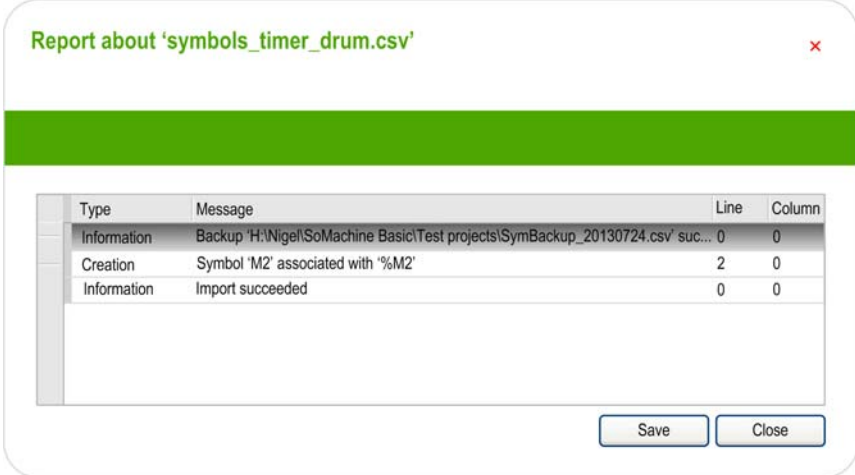
Defining and Using Symbols ([see page 68](#)) describes how to create symbols and use them in your programs.

Displaying the Symbol List

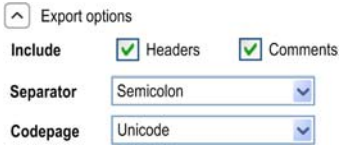
Step	Action
1	Select the Tools tab in the left-hand area of the Programming window.
2	Click Symbol list . Result: The Symbol list window is displayed. For each item the following information is displayed: <ul style="list-style-type: none"> ● Used: Whether the symbol is currently being used in the program. ● Address: The address of the object with which the symbol is associated. ● Symbol: The symbol name. ● Comment: The comment associated with this object, if defined.

Importing Symbols

Step	Action
1	Either click the Import button or right-click anywhere in the list of symbols and choose Import symbols . Result: The Import Symbols window is displayed.
2	Browse and select the File path of the Comma Separated Values (CSV) file containing the symbols to import.
3	Optionally, click Import options and configure formatting options for the imported symbols: 

Step	Action
4	<p>Click Import.</p> <p>Result: All symbols in the selected CSV file are created and displayed in the Symbol list window with the specified formatting options.</p> <p>If errors are detected during import, a report is displayed listing them:</p> 
5	Click Save to write the contents of the report to a plain text (.txt) file.

Exporting the Symbol List

Step	Action
1	<p>Either click the Export button or right-click anywhere in the list of symbols and choose Export symbols. You are prompted to save changes.</p> <p>The Export Symbols window is displayed.</p>
2	Browse and select the File path and File name of the Comma Separated Values (CSV) file to be created.
3	<p>Optionally, click Export options and configure formatting options for the exported values:</p> 
4	<p>Click Export.</p> <p>Result: A CSV file is created with the specified formatting options.</p>

Sharing Symbols Between a SoMachine Basic Project and a Vijeo-Designer Project

Before sharing the symbols with a Vijeo-Designer project, verify that all symbols that you want to share are defined in the SoMachine Basic project. If not, create/open a project in SoMachine Basic, define the symbol names, and save the product.

Follow these steps to share SoMachine Basic symbols with a Vijeo-Designer project:

Step	Action
1	Start Vijeo-Designer.
2	Create/open a project in Vijeo-Designer.
3	Click the Project tab in the Navigator window, right-click IO Manager and select New Driver... Insert . Result: The New Driver window opens.
4	Select a driver from the Driver list, select an equipment from the Equipment list, and click OK . For example: <ul style="list-style-type: none"> ● Driver: Modbus TCP/IP ● Equipment: Modbus Equipment Result: The Equipment Configuration window opens.
5	Enter the details for each parameter and click OK . For example, IP Address , Unit ID , IP Protocol , and so on. Result: A new driver is created to open the communication with the controller. The selected driver and the selected equipment appear under the IO Manager node in the Project tab of the Navigator window.
6	In the Vijeo-Designer menu bar, click Variable → Link Variables . Result: The Link Variable window opens.
7	Select the Files of type filter to SoMachine Basic project files (*.SMBP) and select the Equipment filter to the driver that you have created for communication.
8	Select the SoMachine Basic project in which you have defined the symbols and click Open . Result: All the symbols are automatically extracted from the project and linked to the driver that you have created.
9	Select the variables that you want to use and add them to your HMI application. Result: All the variables with same names as symbols are added in the list of available variables. The variable list appears under the Variables node in the Project tab of the Navigator window.

NOTE: If you have already shared the symbols with a Vijeo-Designer project before and if you change the existing symbols and/or add new symbols to your project in SoMachine Basic, you must update the symbols in the Vijeo-Designer project.

To update the symbols in a Vijeo-Designer project, first define new symbols and/or modify the existing symbols, save the SoMachine Basic project, and open the Vijeo-Designer project and follow these steps:

Step	Action
1	In the Project tab of the Navigator window, right-click Variables and select Update Link . Result: The equipment driver and the existing symbols are updated.
2	Right-click Variables again, select New Variables from Equipment and select the new variables that you have created in the SoMachine Basic project. Result: The new variables from the SoMachine Basic project are added in the list of variables. These variables appear under the Variables node in the Project tab of the Navigator window.

Memory Consumption View

Overview

You can display information about the controller memory used by the application, program, and associated user data.

Displaying the Memory Consumption View


The program must first compile with no errors detected to use this feature. Refer to the Messages window ([see page 111](#)) for the current program status.

To open the **Memory consumption view**, follow this procedure:

Step	Action
1	Select the Tools tab in the left-hand area of the Programming window.
2	Click Memory Consumption . The Memory Consumption window is displayed.

Description of the Memory Consumption View

The following tables describe the fields of the **Memory consumption view**:

Field	Description
Last Compilation	<p>The date and time on which the program was last compiled.</p> <p>NOTE: This value is updated whenever:</p> <ul style="list-style-type: none"> ● the Compile button  on the toolbar is clicked ● a login to a controller is initiated ● a program upload is started ● a program modification is sent to the controller in online mode ● the simulator is launched

Program lines	
Field	Description
Program lines used	The number of lines of code being used by the program.
Program lines remaining	The maximum number of lines available for the program minus the number of lines being used.

Cache memory	
Field	Description
Periodic and Event tasks	The amount of cache memory occupied by periodic and event tasks, in bytes.
Reserved for System	The amount of cache memory reserved for system use, in bytes.
Memory remaining	The amount of available cache memory, in bytes.

RAM memory	
Field	Description
Master task and subroutines	The amount of RAM memory occupied by the program master task and all subroutines, in bytes.
Configuration	The amount of RAM memory used to contain the hardware configuration of the logic controller and expansion modules, in bytes.
Memory objects	The amount of RAM memory occupied by memory objects (memory bits, memory words, and constant words) used by the application, in bytes.
Display	The size of the Remote Graphic Display application, in bytes. Zero if the logic controller does not support the Remote Graphic Display.
Non-program data	The amount of RAM memory occupied by project properties, symbols, and comments, in bytes.
Memory remaining	The amount of RAM memory available to the program, in bytes.

Rung Templates

Overview

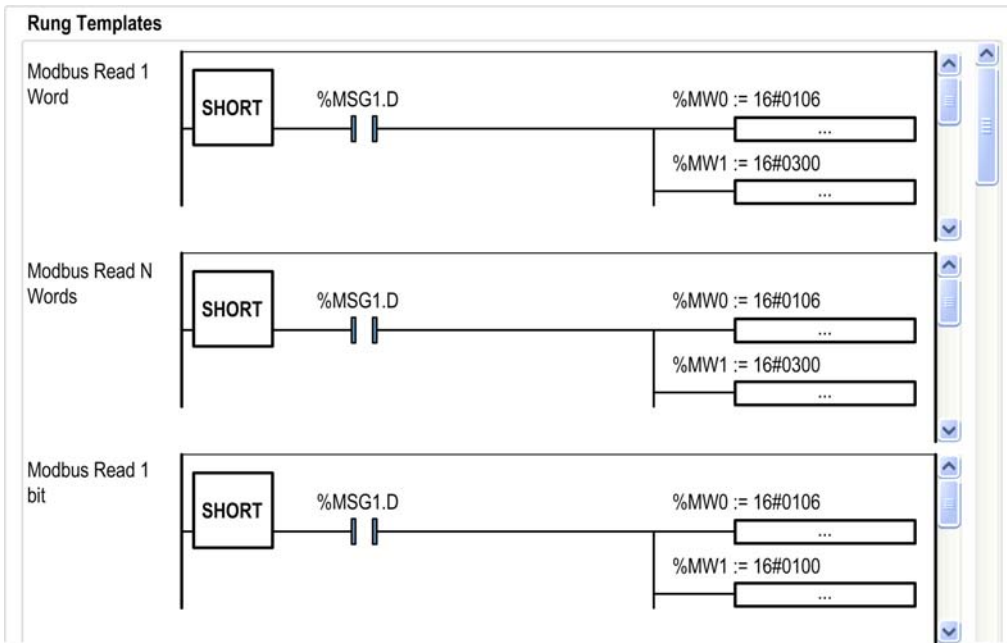
A rung template is a pre-configured portion of source code that you can insert into your programs to make programming quicker while reducing coding errors. SoMachine Basic maintains separate lists of Ladder Diagram and Instruction List rung templates.

Inserting a Rung Template into a Program

Follow these steps to insert a rung template into a program:

Step	Action
1	Select the Tools tab in the left-hand area of the Programming window.
2	Click Rung templates → Ladder or Rung templates → Instruction List . A list of current rung templates in Ladder Diagram or Instruction List format is displayed.
3	<p>Inserting a rung template into a program is possible by any of the following methods:</p> <ul style="list-style-type: none"> ● Select a rung in your program in the programming workspace and then double-click a rung template. ● Right-click on a rung template and click Copy rung in the context menu and then right-click in the programming workspace and click Paste Rung in the context menu. <p>Result: The rung template is always inserted after the last rung in a POU. Use UP and DOWN arrow button on the toolbar at the top of the Tasks tab to reposition the rungs in your program.</p>

This graphic shows the rung templates in Ladder Diagram language:



This graphic shows the rung templates in Instruction List language:

Rung Templates

Modbus Read 1 Word	0000	LD 1
	0001	AND %MSG1.D
	0002	[%MW0 := 16#0106]
	0003	[%MW1 := 16#0300]
	0004	[%MW2 := 1]
Modbus Read N Words	0000	LD 1
	0001	AND %MSG1.D
	0002	[%MW0 := 16#0106]
	0003	[%MW1 := 16#0300]
	0004	[%MW2 := 1]
Modbus Read 1 bit	0000	LD 1
	0001	AND %MSG1.D
	0002	[%MW0 := 16#0106]
	0003	[%MW1 := 16#0100]
	0004	[%MW2 := 1]

Section 6.9

Ladder Language Programming

What Is in This Section?



This section contains the following topics:

Topic	Page
Introduction to Ladder Diagrams	135
Programming Principles for Ladder Diagrams	137
Ladder Diagram Graphic Elements	138
Comparison Blocks	144
Operation Blocks	145
Adding Comments	146
Programming Best Practices	147

Introduction to Ladder Diagrams

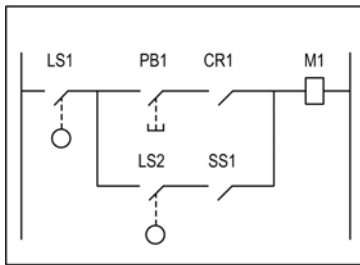
Introduction

Ladder Diagrams are similar to relay logic diagrams that represent relay control circuits. The main differences between the 2 are the following features of Ladder Diagram programming that are not found in relay logic diagrams:

- All inputs and binary logic bits are represented by contact symbols ().
- All outputs and binary logic bits are represented by coil symbols ().
- Numerical operations are included in the graphical Ladder instruction set.

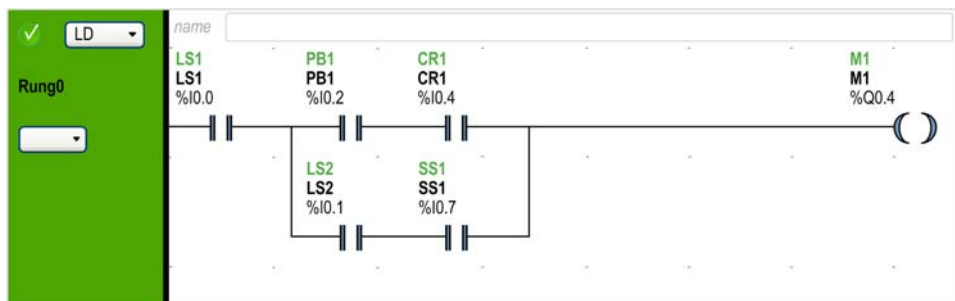
Ladder Diagram Equivalents to Relay Circuits

The following illustration shows a simplified wiring diagram of a relay logic circuit:



Relay logic circuit

The equivalent Ladder diagram:



In the above illustration, all inputs associated with a switching device in the relay logic diagram are shown as contacts in the Ladder Diagram. The M1 output coil in the relay logic diagram is represented with an output coil symbol in the Ladder Diagram. The address numbers appearing above each contact/coil symbol in the Ladder Diagram are references to the locations of the external input/output connections to the logic controller.

Ladder Diagram Rungs

A program written in Ladder Diagram language is composed of rungs which are sets of graphical instructions drawn between 2 vertical potential bars. The rungs are executed sequentially by the logic controller.

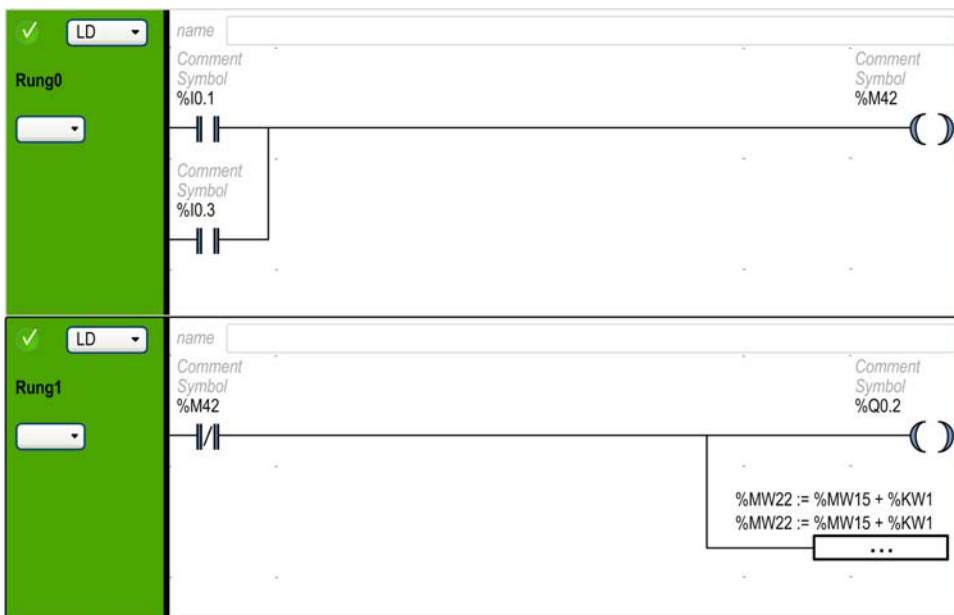
The set of graphical instructions represents the following functions:

- Inputs/outputs of the controller (push buttons, sensors, relays, pilot lights, and so on)
- Functions of the controller (timers, counters, and so on)
- Math and logic operations (addition, division, AND, XOR, and so on)
- Comparison operators and other numerical operations ($A < B$, $A = B$, shift, rotate, and so on)
- Internal variables in the controller (bits, words, and so on)

These graphical instructions are arranged with vertical and horizontal connections leading eventually to one or several outputs and/or actions. A rung cannot support more than one group of linked instructions.

Example of Ladder Diagram Rungs

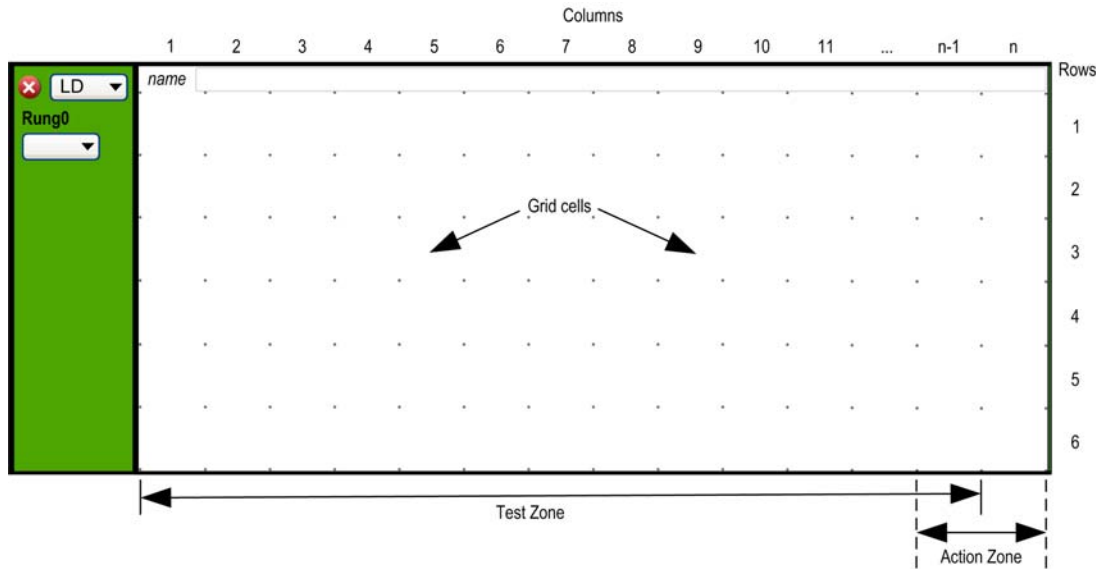
The following diagram is an example of a Ladder Diagram program composed of 2 rungs.



Programming Principles for Ladder Diagrams

Programming Grid

Each Ladder rung consists of a grid of up to 1,000 rows by 11...30 columns that are organized into 2 zones as shown in the following illustration:



n Number of configured columns (11...30). For more information on configuration of number of columns, refer to Customizing the Ladder Editor ([see page 49](#)).

Grid Cells

Cells allow you to position graphical elements in the grid. Each cell in the grid is delimited by 4 dots at the corners of the cell.

Grid Zones

By default, the Ladder Diagram programming grid is divided into 2 zones:

- Test zone
Contains the conditions that are tested in order to perform actions. Consists of columns 1 to n-1, where n is the number of configured columns and contains contacts, function blocks, and comparison blocks.
- Action zone
Contains the output or operation that will be performed according to the results of the tests of the conditions in the Test zone. Consists of columns n-1 to n, where n is the number of configured columns and contains coils and operation blocks.


Ladder Diagram Graphic Elements

Introduction

Instructions in Ladder Diagrams are inserted by dragging and dropping graphic elements from the toolbar that appears above the programming workspace into a grid cell.




Inserting a Graphic Element

To insert a graphic element in a rung:

Step	Action
1	Click the graphic element on the toolbar to insert. If the graphic element is a menu, the graphic items in the menu appear; click the menu item to insert.
2	Move the mouse to the position in the rung to insert the graphic element and click. Note: Some elements have to be inserted in the test or action zones of the rung; refer to the description of individual graphic elements for details.
3	If necessary, click the [Selection mode] graphic element  on the toolbar to reset the selection.



Rungs

Use the following graphic elements to manage the rungs in a program:

Graphic Element	Name	Function
	Create a rung <i>(see page 90)</i>	Inserts a new empty rung below the last rung in the program workspace.
	Insert a rung <i>(see page 90)</i>	Inserts a new empty rung immediately above the currently selected rung.
	Delete the rung <i>(see page 92)</i>	Removes the currently selected rung from the program. If the rung is not empty, you are asked to confirm that you want to delete the contents of the rung.




Branching Modes

Use the following graphic elements to manage the branch in Ladder diagram:

Graphic Element	Name	Function
	Normal mode	Lets you place the programming elements (for example, contacts, coils, and so on, except the function blocks) inline with the wire line.
	Branching mode	Lets you place the programming elements (for example, contacts, coils, and so on, except the function blocks) in branch with the wire line.





Selections and Lines

Use the following graphic elements to select graphic elements and draw lines:

Graphic Element	Name	Function
	Selection mode	Selection mode.
	Draw line	Draws a wire line between 2 graphic elements.
	Erase line	Erases a wire line.

Contacts


Use the following graphic elements to insert contacts (one row high by one column wide).

Graphic Element	Name	Instruction List	Function
	Normally open contact	LD	Passing contact when the controlling bit object is at state 1.
	Normally closed contact	LDN	Passing contact when the controlling bit object is at state 0.
	Contact for detecting a rising edge	LDR	Rising edge: detecting the change from 0 to 1 of the controlling bit object.
	Contact for detecting a falling edge	LDF	Falling edge: detecting the change from 1 to 0 of the controlling bit object.

Comparison Block

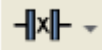
Comparison blocks are placed in the test zone of the programming grid. The block may appear in any row or column in the test zone as long as the entire length of the instruction resides in the test zone.

The graphic element for comparison blocks takes up 2 cells (1 row high by 2 columns wide).

Graphic Element	Name	Instruction List	Function
	Comparison block	Any valid comparison expression	Use the Comparison block graphical symbol to insert Instruction List comparison expressions (<i>see page 144</i>) into Ladder Diagram rungs. A comparison expression compares 2 operands; the output changes to 1 when the result is checked.

Boolean Operations


The graphic element for boolean operations takes up 1 cell (1 row high by 1 column wide).

Graphic Element	Name	Operator	Function
	XOR instructions	XOR, XORN, XORR, XORF	The XOR instruction performs an exclusive OR operation between the operand and the Boolean result of the preceding instruction. The XORN instruction performs an exclusive OR operation between the inverse of the operand and the Boolean result of the preceding instruction. The XORR instruction performs an exclusive OR operation between the rising edge of the operand and the Boolean result of the preceding instruction. The XORF instruction performs an exclusive OR operation between the falling edge of the operand and the Boolean result of the preceding instruction.

Functions





Function blocks always appear in the first row of the Ladder Diagram programming grid; no Ladder instructions or lines of continuity may appear above or below the function block. Ladder test instructions lead to the left side of the function block, and test instructions and action instructions lead from the right side of the function.

The graphic elements of function blocks can only be placed in the test zone and require 2, 3, or 4 rows by 2 columns of cells.

Graphic Element	Name	Function
	Timers, counters, registers, and so on.	Each of the function blocks uses inputs and outputs that enable links to the other graphic elements. NOTE: Outputs of function blocks cannot be connected to each other (vertical shorts).



Coils

The coil graphic elements can only be placed in the action zone and take up 1 cell (1 row high and 1 column wide).

Graphic Element	Name	Operator	Function
	Direct coil	ST	The associated bit object takes the value of the test zone result.
	Inverse coil	STN	The associated bit object takes the negated value of the test zone result.
	Set coil	S	The associated bit object is set to 1 when the result of the test zone is 1.
	Reset coil	R	The associated bit object is set to 0 when the result of the test zone is 1.


Grafcet Instructions

Use the following graphic elements to manage the branch in Ladder diagram:

Graphic Element	Name	Operator	Function
	Grafcet step activation/ Current step deactivation	#	Deactivates the current step and optionally activates another step in the Grafcet program.
	Grafcet step deactivation	#D	Deactivates a step in the Grafcet program in addition to deactivating the current step.

Operation Blocks

The operation block element placed in the action zone and occupy 2 columns by 1 row:



Graphic Element	Name	Operator	Function
	Operation block	Any valid operator or assignment instruction	Use the Operation block graphical symbol to insert Instruction List operations and assignment instructions (<i>see page 145</i>) into Ladder Diagram rungs.






Other Items

The **Other Items** menu  groups together miscellaneous instructions.

The **OPEN** and **SHORT** instructions provide a convenient method for debugging and troubleshooting Ladder programs. These special instructions alter the logic of a rung by either shorting or opening the continuity of a rung as explained in the following table.

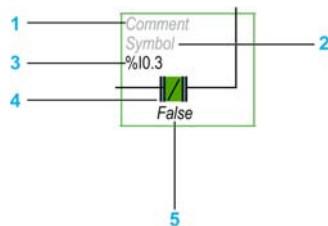
The **END/JUMP** graphic elements are placed in the action zone and take up 1 cell (1 row high and 1 column wide).

Graphic Element	Name	Operator	Function
	Logical NOT	N	Passes the inverse value of its operand.
	OPEN	LD 0 AND 0	At the beginning of the rung. Within a rung: Creates a break in the continuity of a Ladder rung regardless of the results of the last logical operation.

Graphic Element	Name	Operator	Function
	SHORT	LD 1 OR 1	At the beginning of the rung. Within a rung: Allows the continuity to pass through the rung regardless of the results of the last logical operation.
	Stop program	END	Defines the end of the program.
	Conditional stop program	ENDCN	Defines a conditional end of the program.
	Jump or subroutine call	JMP	Connect to an upstream or downstream labeled rung. NOTE: When programming in IL, connection is to an upstream or downstream labeled instruction.
	Conditional jump or subroutine call	JMPCN	Conditional connect to an upstream or downstream labeled rung. NOTE: When programming in IL, connection is to an upstream or downstream labeled instruction.

Contacts and Coils

Once inserted in a cell, additional information is displayed about the object associated with contacts and coils:



Legend	Item	Description
1	User comment	Click to add a comment (see page 146).
2	Symbol	Click to type the name of a symbol (see page 68) to associate with the object contained in the cell.
3	Address	Click to type the address of the object contained in the cell.
4	Graphic element	The graphic element.
5	Real-time value	When in online mode (connected to a logic controller and program running), displays the real-time value of the object in the cell.


Comparison Blocks

Inserting IL Comparison Expressions in Ladder Diagrams

You can use the **Comparison Block** graphical symbol to insert Instruction List comparison expressions into Ladder Diagram rungs:



Proceed as follows:

Step	Action
1	Click the Comparison Block  button on the toolbar.
2	Click anywhere in the rung to insert the Comparison Block .
3	Double-click the Comparison expression line.
4	Type a valid Instruction List comparison operation and press ENTER.

Getting Help with Syntax

If the syntax of the Instruction List comparison operation is incorrect, the border of the **Comparison expression** box turns red. For assistance, either:

- Move the mouse over the **Comparison expression** line, or
- Select **Tools** → **Program Messages**.


Operation Blocks

Inserting IL Operations and Assignment Instructions in Ladder Diagrams

You can use the **Operation Block** graphical symbol to insert Instruction List operations and assignment instructions into Ladder Diagram rungs:



To insert an operation block in a rung:

Step	Action
1	Click the Operation Block  button on the toolbar.
2	Click in the Action zone (last 2 columns) of the rung to insert the Operation Block .
3	Double-click the operation expression line.
4	Type a valid Instruction List operation or assignment instruction and press ENTER.

Getting Help with Syntax

If the syntax of the Instruction List operation or assignment instruction is incorrect, the border of the **operation expression** box turns red. For assistance, either:

- Move the mouse over the **operation expression** line, or
- Select **Tools** → **Program Messages**.

Adding Comments

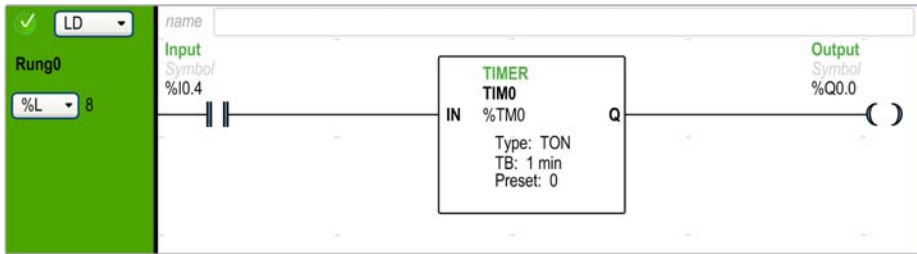
To Add Comments to Ladder Diagrams

To add comments to a Ladder Diagram program, follow these steps:

Step	Action
1	Insert a graphic element into the rung.
2	If necessary, click the selection pointer or press Esc.
3	Double-click the Comment line at the top of the graphic element.
4	Type the comment for the graphic element and press ENTER.

Example of Ladder Diagram Comments

This illustration shows an example of comments in a rung of a Ladder Diagram:



Programming Best Practices

Handling Program Jumps

Use program jumps with care to avoid long loops that can increase scan time. Avoid jumps to instructions that are located upstream.

NOTE: An upstream instruction line appears before a jump in a program. A downstream instruction line appears after a jump in a program.

Programming of Outputs

Physical outputs, as well as logical bits, should only be modified once in the program. In the case of physical outputs, only the last value scanned is taken into account when they are updated.

Using Directly-Wired Emergency Stop Sensors

Sensors used directly for emergency stops must not be processed by the logic controller. They must be connected directly to the corresponding outputs and applied in conformity with local, national and/or international regulations.

Handling Power Returns

After a power outage, make power returns conditional on a manual operation. An automatic restart of the installation could cause unexpected operation of equipment (use system bits %S0, %S1, and %S9).

WARNING

UNINTENDED EQUIPMENT OPERATION

Do not use the equipment configured and programmed by this software in safety-critical machine functions, unless the equipment and software are otherwise designated as functional safety equipment and conforming to applicable regulations and standards.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Time and Schedule Block Management

The state of system bit %S51, which indicates any detected RTC errors should be verified.

Syntax Validation

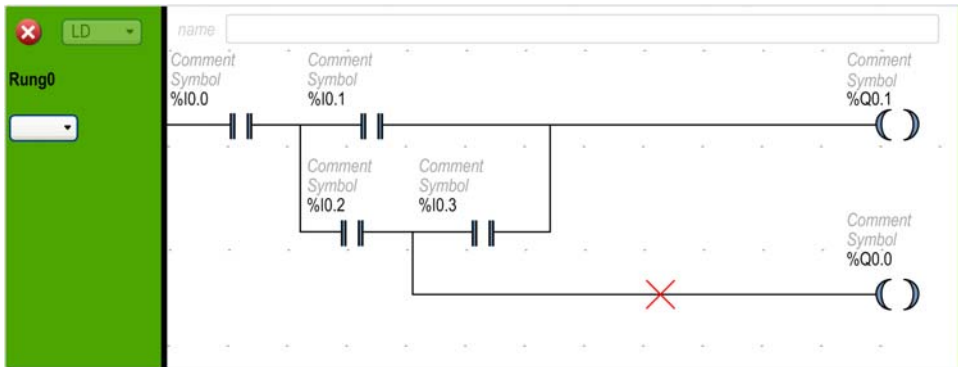
As you are programming, SoMachine Basic validates the syntax of the instructions, the operands, and their associations.

Additional Notes on Using Parentheses

Do not place assignment instructions within parentheses:

```
LD    %I0.0
MPS
AND  %I0.1
OR (  %I0.2
)
```

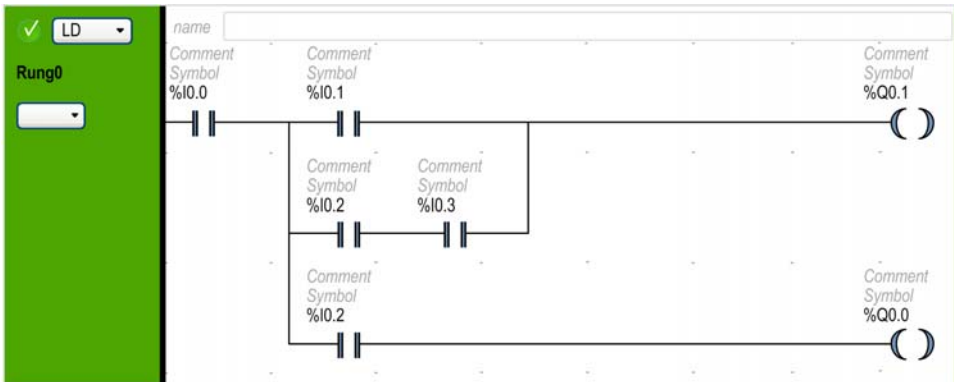
The equivalent Ladder Diagram produces a short circuit error:



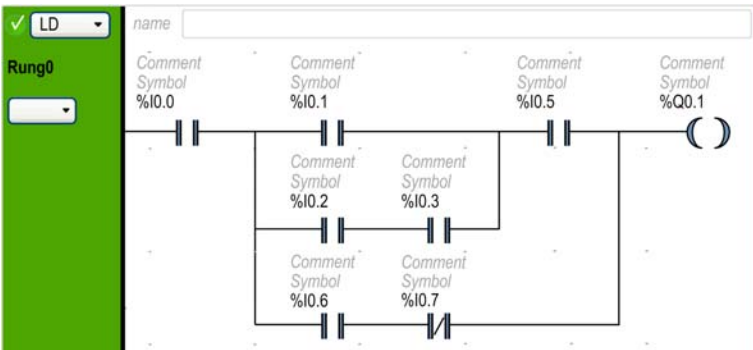
In order to perform the same function, program the instructions in the following manner:

```
LD    %I0.0
MPS
AND ( %I0.1
OR (  %I0.2
AND  %I0.3
)
)
ST   %Q0.1
MPP
AND  %I0.2
ST   %Q0.0
```

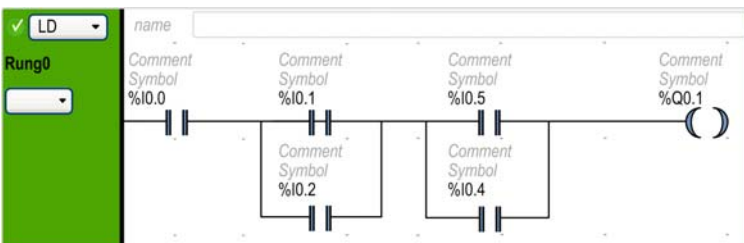
The equivalent Ladder Diagram:



If several contacts are in parallel, nest them within each other:



Alternatively, completely separate the contacts as follows:



Section 6.10

Instruction List Programming

What Is in This Section?

This section contains the following topics:

Topic	Page
Overview of Instruction List Programs	151
Operation of List Instructions	153
List Language Instructions	154
Using Parentheses	158

Overview of Instruction List Programs

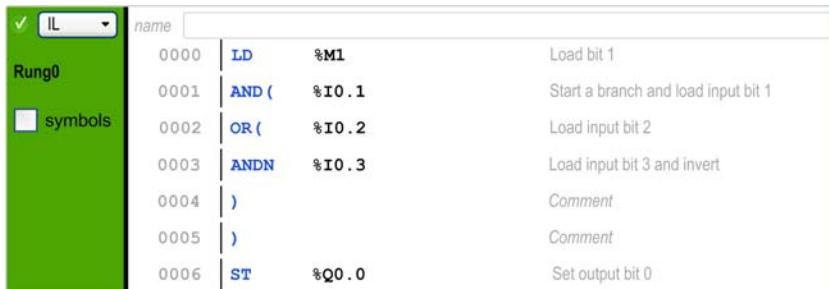
Introduction

A program written in Instruction List language consists of a series of instructions that are executed sequentially by the logic controller. Each instruction is represented by a single program line and consists of the following components:

- Line number
- Current value (in online mode only)
- Instruction operator
- Operand(s)
- Optional comment

Example of an Instruction List Program

The following is an example of an Instruction List program.



Line Number	Instruction	Operand	Comment
0000	LD	%M1	Load bit 1
0001	AND (%I0.1	Start a branch and load input bit 1
0002	OR (%I0.2	Load input bit 2
0003	ANDN	%I0.3	Load input bit 3 and invert
0004)		Comment
0005)		Comment
0006	ST	%Q0.0	Set output bit 0

Line Numbers

Four-digit line numbers are generated when you create a new program line and managed automatically by SoMachine Basic.

Current Values

When SoMachine Basic is in online mode ([see page 27](#)) (connected to a logic controller and the program is running), SoMachine Basic displays the current value of object types in the IL editor window.

The displayed values of these objects are updated.

Instruction Operators

The instruction operator is a mnemonic symbol, called an operator, that identifies the operation to be performed using the operands. Typical operators specify Boolean and numerical operations.

For example, in the sample program above, `LD` is the mnemonic for the `LOAD` operator. The `LOAD` instruction places (loads) the value of the operand `%M1` into an internal register called the boolean accumulator.

There are basically 2 types of operators:

- Test operators
These set up or test for the conditions necessary to perform an action. For example, `LOAD (LD)` and `AND`.
- Action operators
These perform actions as a result of preceding logic. For example, assignment operators such as `STORE (ST)` and `RESET (R)`.

Operators, together with operands, form instructions.

Operands

An operand is an object, address, or symbol representing a value that a program can manipulate in an instruction. For example, in the sample program above, the operand `%M1` is an address assigned the value of an embedded input of the logic controller. An instruction can have from 0 to 3 operands depending on the type of instruction operator.

Operands can represent the following:

- Controller inputs and outputs such as sensors, push buttons, and relays.
- Predefined system functions such as timers and counters.
- Arithmetic, logical, comparison, and numerical operations.
- Controller internal variables such as system bits and words.

Comments

To add comments to an Instruction List program

Step	Action
1	Optionally, click the comment box that appears at the top of the rung above the first line 0000 and type a comment for the rung.
2	Insert an instruction line.
3	Click in the Comment area to the right of the instruction.
4	Type the comment and press <code>Enter</code> .

Operation of List Instructions

Introduction

Instruction List binary instructions normally have only one explicit operand; the other operand is implied. The implied operand is the value in the Boolean accumulator. For example, in the instruction `LD %I0.1`, `%I0.1` is the explicit operand. An implicit operand is loaded in the accumulator and the previous value of the accumulator is overwritten by the value of `%I0.1`. This value now becomes the implicit value for the subsequent instruction.

Operation

An Instruction List instruction performs a specified operation on the contents of the accumulator and the explicit operand, and replaces the contents of the accumulator with the result. For example, the operation `AND %I1.2` performs a logical AND between the contents of the accumulator and the input `1.2` and will replace the contents of the accumulator with this result.

All Boolean instructions, except for `Load`, `Store`, and `Not`, operate on 2 operands. The value of the 2 operands can be either True or False, and program execution of the instructions produces a single value: either True or False. `Load` instructions place the value of the operand in the accumulator while `Store` instructions transfer the value in the accumulator to the operand. The `Not` instruction has no explicit operands and simply inverts the state of the accumulator.

Supported List Instructions

This table shows a selection of instructions in Instruction List language:

Type of Instruction	Example	Function
Boolean instruction	<code>LD %M10</code>	Loads the value of internal bit <code>%M10</code> into the accumulator
Block instruction	<code>IN %TM0</code>	Starts the timer <code>%TM0</code>
Word instruction	<code>[%MW10 := %MW50+100]</code>	Addition operation
Program instruction	<code>SR5</code>	Calls subroutine #5

List Language Instructions

Introduction







The Instruction List language consists of the following types of instructions or block of instructions:


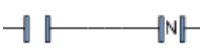
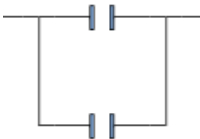
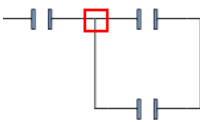
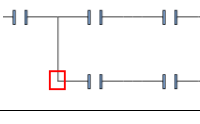
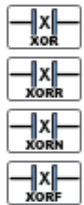
- Test Instructions
- Action instructions
- Function blocks

This section identifies and describes the instructions for List programming.

Test Instructions

This table describes test instructions in List language.

Mnemonic	Name	Equivalent Graphic Element	Function
LD	Load		Loads the boolean value of the operand into the accumulator.
LDN	Load Not		Loads the negated boolean value of the operand into the accumulator.
LDR	Load Rising		Loads the boolean value of the operand into the accumulator when the value changes from 0 to 1 (rising edge). The value of the accumulator thereafter will be loaded with 0 until the next transition of the operand from 0 to 1.
LDF	Load Falling		Loads the boolean value of the operand into the accumulator when the value changes from 1 to 0 (falling edge). The value of the accumulator thereafter will be loaded with 1 until the next transition of the operand from 1 to 0.
AND	And		The Boolean result is equal to the AND logic between the Boolean result of the previous instruction (which is stored in the accumulator) and the status of the operand. The result of the instruction is then itself implicitly loaded into the accumulator overwriting the previous value.
ANDN	And Not		The Boolean result is equal to the AND logic between the Boolean result of the previous instruction (which is stored in the accumulator) and the inverse (negated) status of the operand. The result of the instruction is then itself implicitly loaded into the accumulator overwriting the previous value.

Mnemonic	Name	Equivalent Graphic Element	Function
ANDR	And Rising		The Boolean result is equal to the AND logic between the Boolean result of the previous instruction and the detection of the operand's rising edge (1 = rising edge). The result of the instruction is then itself implicitly loaded into the accumulator overwriting the previous value.
ANDF	And Falling		The Boolean result is equal to the AND logic between the Boolean result of the previous instruction and the detection of the operand's falling edge (1 = falling edge). The result of the instruction is then itself implicitly loaded into the accumulator overwriting the previous value.
OR	Or		The Boolean result is equal to the OR logic between the Boolean result of the previous instruction and the status of the operand (which is stored in the accumulator).
AND(And With		Logic AND (Maximum 32 levels of parentheses). The parentheses specify an intermediate logical result of the instructions between them, and then that result is logically AND'd with the value in the accumulator.
OR(Or With		Logic OR (Maximum 32 levels of parentheses). The parentheses specify an intermediate logical result of the instructions between them, and then that result is logically OR'd with the value in the accumulator.
XOR XORN XORR XORF	Ex Or Ex Or Not Ex Or Rising Ex Or Falling		Exclusive OR

Mnemonic	Name	Equivalent Graphic Element	Function
MPS MRD MPP	Memory Push Store Memory ReaD Memory PoP		Branch operators for output actions.
N	Not		Inverts the value of the operand.


Action Instructions

This table describes action instructions in List language.

Mnemonic	Name	Equivalent Graphic Element	Function
ST	Store		The associated operand takes the value of the test zone result.
STN	Store Not		The associated operand takes the reverse value of the test zone result.
S	Set		The associated operand is set to 1 when the result of the test zone is 1.
R	Reset		The associated operand is set to 0 when the result of the test zone is 1.
JMP	Jump		Connect unconditionally to a labeled sequence, upstream, or downstream.
SRn	Subroutine		Connection at the beginning of a subroutine (subroutine call).
END	End		End of program.
ENDCN	End Conditional		Conditionally ends the program at a Boolean result of 0.

Function Blocks

This table describes function blocks in List language.

Name	Equivalent Graphic Element	Function
Timers, counters, registers, and so on.		<p>For each of the function blocks, there are instructions for controlling the block.</p> <p>A structured form is used to connect the block inputs and outputs.</p> <p>Note: Outputs of function blocks cannot be connected to each other (vertical shorts).</p> <p>For more information, refer to Software Objects (see <i>SoMachine Basic, Generic Functions Library Guide</i>).</p>

Using Parentheses

Introduction

With **AND** and **OR** logical operators, parentheses are used to nest logical instructions. In so doing, they specify divergences (branches) in the Ladder editor. Parentheses are associated with instructions as follows:

- Opening the parentheses is associated with the **AND** or **OR** operator.
- Closing the parentheses is an instruction (an operator with no operand) which is required for each open parenthesis.

Example Using an **AND** Instruction

The following examples show how to use parentheses with an **AND** instruction:

Rung	Instruction
0	LD %I0.0 AND %I0.1 OR %I0.2 ST %Q0.0
1	LD %I0.0 AND(AND %I0.1 OR %I0.2) ST %Q0.1

NOTE: Refer to the reversibility procedure ([see page 76](#)) to obtain the equivalent Ladder Diagram.

Example Using an **OR** Instruction

The following example shows how to use parentheses with an **OR** instruction:

Rung	Instruction
0	LD %I0.0 AND %I0.1 OR(AND %I0.3) ST %Q0.0

NOTE: Refer to the reversibility procedure ([see page 76](#)) to obtain the equivalent Ladder Diagram.

Modifiers

This table lists modifiers that can be assigned to parentheses.

Modifier	Function	Example
N	Negation	AND(N or OR(N
F	Falling edge	AND(F or OR(F
R	Rising edge	AND(R or OR(R
[Comparison	See Comparison Instructions.

NOTE: The '[' modifier can also be used in conjunction with other instructions serving as an operator. For more uses of the '[' in other instructions, refer to the Introduction to Numerical Operations.

Nesting Parenthesis

It is possible to nest up to 32 levels of parentheses.

Observe the following rules when nesting parentheses:

- Each open parenthesis must have a corresponding closed parenthesis.
- Labels (%Li:), subroutines (SRi:), JMP instructions (JMP), and function block instructions must not be placed in expressions between parentheses.
- Store instructions (ST, STN, S, and R) must not be programmed between parentheses.
- Stack instructions (MPS, MRD, and MPP) cannot be used between parentheses.

Examples of Nesting Parentheses

The following examples show how to nest parentheses:

Rung	Instruction
0	LD %I0.0 AND(%i0.1 OR(N %i0.2 AND %M3)) ST %Q0.0

Rung	Instruction
1	<pre> LD %I0.1 AND(%I0.2 OR(%I0.5 AND %I0.6) AND %I0.4 OR(%I0.7 AND %I0.8)) ST %Q0.0 </pre>

NOTE: Refer to the reversibility procedure ([see page 76](#)) to obtain the equivalent Ladder Diagram.

Section 6.11

Grafcet (List) Programming

What Is in This Section?

This section contains the following topics:

Topic	Page
Description of Grafcet (List) Programming	162
Grafcet Program Structure	163
How to Use Grafcet Instructions in a SoMachine Basic Program	167

Description of Grafcet (List) Programming




Introduction

Grafcet (List) programming in SoMachine Basic offer a simple method of translating a control sequence in to steps. You can translate control sequences in to Grafcet steps and then use these steps in a program using Grafcet instructions.

The maximum number of Grafcet steps depend on the controller. The number of steps active at any one time is limited only by the total number of steps.

Grafcet Instructions

A SoMachine Basic Grafcet program has the following instructions:

Operator	Operand	IL Instruction	Instruction Name	Graphic Equivalent	Description
=*=	x	=*= x	INITIAL STEP		This instruction defines the initial step in the program.
=*= POST	Not applicable	=*= POST	POST PROCESSING (implicit operand)		This instruction defines the post-processing and end sequential processing.
-*-	x	-*- x	STEP		This instruction defines a step in the program for transition validation.
#	Not applicable	#	DEACTIVATE CURRENT STEP (implicit operand)		This instruction deactivates the current step in the program.
#	x	# x	DEACTIVATE CURRENT STEP and ACTIVATE STEPx		This instruction deactivates the current step and activates step x in the program.
#D	x	#D x	DEACTIVATE CURRENT STEP and STEPx		This instruction deactivates the current step and step x in the program.

x Grafcet step number (an integer starting from 1).

Grafcet Program Structure

Introduction

A SoMachine Basic Grafcet program has the following parts:

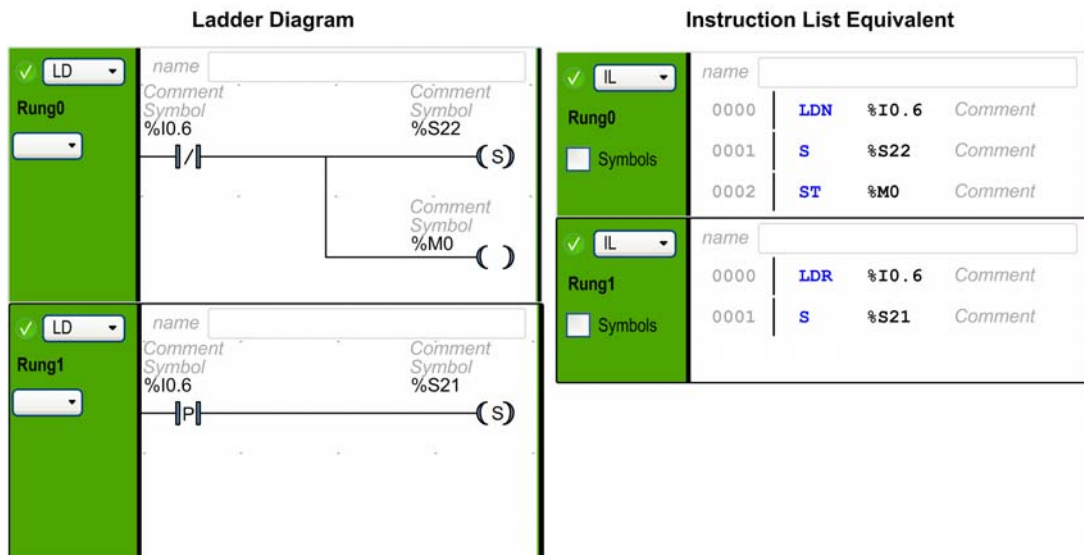
- Preprocessing
- Sequential processing
- Post-Processing

Preprocessing

Preprocessing consists of the following:

- Power returns
- Error management
- Changes of operating mode
- Pre-positioning Grafcet steps
- Input logic

In this example, the system bit %S21 to 1 with the rising edge of input %I0.6. This disables the active steps and enables the initial steps:



Preprocessing begins with the first line of the program and ends with the first occurrence of a =* or -* instruction.

System bits %S21, %S22, and %S23 are dedicated to Grafcet control. Each of these system bits is set to 1 (if needed) by the application, normally in preprocessing. The associated function is performed by the system at the end of preprocessing and the system bit is then reset to 0 by the system.

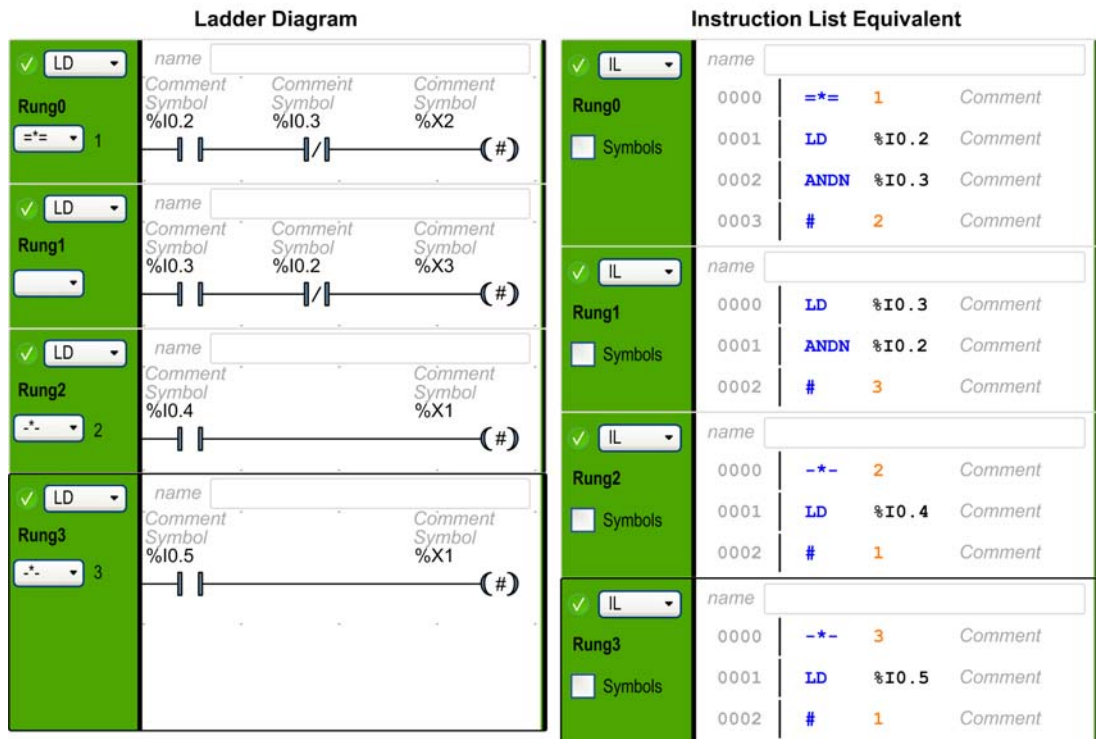
System Bit	Name	Description
%S21	Grafcet initialization	All active steps are deactivated and the initial steps are activated.
%S22	Grafcet re-initialization	All steps are deactivated.
%S23	Grafcet pre-positioning	This bit must be set to 1 if %Xi objects are explicitly written by the application in preprocessing. If this bit is maintained to 1 by the preprocessing without any explicit change of the %Xi objects, Grafcet is frozen (no updates are taken into account).

Sequential Processing

Sequential processing takes place in the chart (instructions representing the chart):

- Steps
- Actions associated with steps
- Transitions
- Transition conditions

Example:



Sequential processing ends with the execution of the **POST** instruction or with the end of the program.

Post-Processing

Post-processing consists of the following:

- Commands from the sequential processing for controlling the outputs
- Interlocks specific to the outputs

Example:

Ladder Diagram

The Ladder Diagram consists of three rungs. Rung 0 starts with a normally closed contact labeled '%I0.2', followed by a normally open contact labeled '%I0.3', and ends with a coil labeled '%X2'. Rung 1 starts with a normally open contact labeled '%I0.4' and ends with a coil labeled '%Q0.2'. Rung 2 starts with a normally open contact labeled '%X1' and ends with a coil labeled '%Q0.1'.

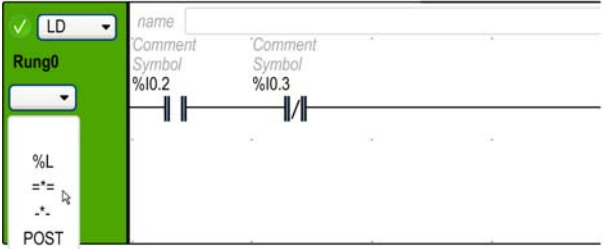
Instruction List Equivalent

IL	name			
0000	==	1		Comment
0001	LD	%I0.2		Comment
0002	ANDN	%I0.3		Comment
0003	#	2		Comment
0000	--	2		Comment
0001	LD	%I0.4		Comment
0002	ST	%Q0.2		Comment
0000	==	POST		Comment
0001	LD	%X1		Comment
0002	ST	%Q0.1		Comment

How to Use Grafcet Instructions in a SoMachine Basic Program



Creating Grafcet Steps in Ladder

Follow these steps to create Grafcet steps in a program:

Step	Action
1	<p>In a POU, select a rung and click the drop-down button below the rung sequence identifier Rungx, where x is the rung number in a POU.</p>  <p>Result: A menu appears listing the Grafcet instructions.</p>
2	<p>Click an instruction in the list to define the rung as an initial step, post processing, or a step of the Grafcet program.</p> <p>Result: The rung is set for a Grafcet instruction. The operator of the instruction appears on the button and the operand (step number) appears in suffix with the button.</p> <p>NOTE: The step number is incremented by 1 as you define the next STEP or INITIAL STEP instruction. You can define only one POST instruction in a program; therefore the POST instruction does not have any step number.</p> <p>To modify the step number, double-click the step number in a rung and enter the new number and then press ENTER.</p>

Activating or Deactivating Grafcet Steps in Ladder

Follow these steps to activate or deactivate Grafcet steps in a program:

Step	Action
1	In a POU, select a rung in your program.
2	 <p>Click (to deactivate the current step and optionally activate a specified step) or  (to deactivate the current step and to deactivate the specified step) and insert this element in the action zone of the rung (refer to Inserting a Graphic Element (see page 138)).</p>
3	<p>Alternatively, Press ALT+A to use ACTIVATE instruction or press ALT+D to use DEACTIVATE instruction in the rung.</p> <p>Result: The activate or deactivate ladder symbol appears in the action zone of the rung. Press ENTER to insert this element.</p>

Step	Action
4	<p>In the program rung, double-click Address field on the Grafcet activate or deactivate symbol and enter the Grafcet bit address (%Xi, where i is the step number).</p> <p>For example, %X4 refers to the step 4 of the Grafcet program. If %X4 is the address for the deactivate symbol, step 4 will be deactivated when the output of the rung, in which this symbol is used, is true.</p> <p>NOTE: Current step is deactivated in every case.</p>

Section 6.12

Debugging in Online Mode

What Is in This Section?

This section contains the following topics:

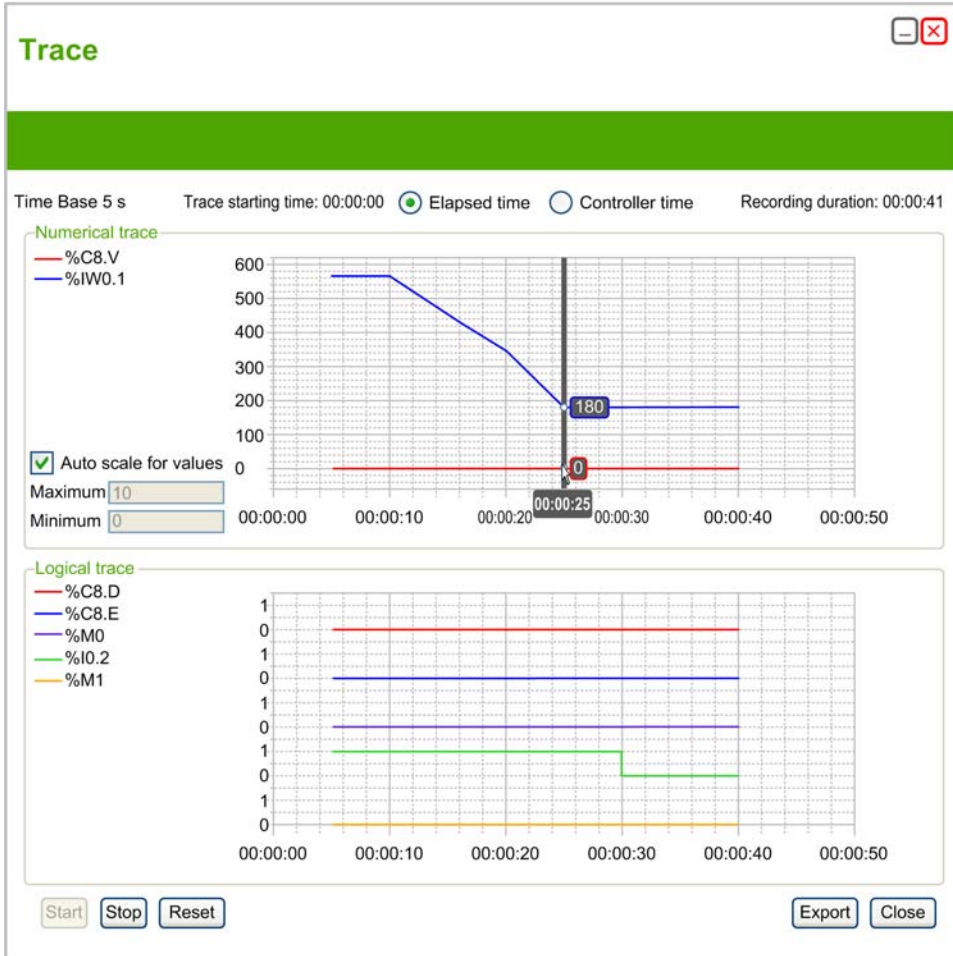
Topic	Page
Trace Window	170
Modifying Values	172
Forcing Values	173
Online Mode Modifications	174

Trace Window

Overview

The **Trace** window allows you to display in graphical form the values of specific analog and/or digital variables. You can export the data to a file for further analysis.

Trace Window Presentation



Select **Elapsed time** to set tracing start time to 00:00:00, or **Controller time** to use the time and date of the logic controller as the start time of the trace.

The Trace window displays separate graphs for each data type selected for tracing in the animation table:

- Integer and real values appear in the **Numerical trace** area.
All numerical values share the same scale on the graph.
Select **Auto scale for values** to automatically adjust the vertical axis to display all values.
Otherwise, type **Maximum** and **Minimum** values to display a fixed range of values.
NOTE: You can type either integer or real values for **Maximum** and **Minimum**
- Binary values appear in the **Logical trace** area.
Each binary value is traced on an individual scale of 0 and 1.

Starting, Pausing, and Resetting the Trace

Click **Start** to begin tracing the variables.

Click **Stop** to pause real-time tracing.

Click **Reset** to clear all previously traced data from graphs and reset the **Recording duration** value to 0.

Exporting the Trace

Click **Export** to export all traced data to a file on the PC.

The data is saved in comma-separated value (CSV) format.

Modifying Values

Introduction

When in online mode, SoMachine Basic allows you to modify the values of certain object types.

Online updating is only possible if the object has read/write access. For example:

- The value of an analog input cannot be modified.
- The value of the `Preset` parameter (`%TM0.P` object) of a `Timer` function block can be updated.

Refer to the description of objects in the SoMachine Basic *Generic Functions Library Guide* or the *Programming Guide* of your hardware platform for information on which object types have read/write access.

To modify the value of an object, add it to an animation table ([see page 114](#)) and set its properties as required.

Forcing Values

Overview

When in online mode, you can force the values of digital inputs and outputs to False (0) or True (1). This allows you to set addresses to specific values and prevent the program logic or an external system from changing the value. This function is mainly used for the debugging and fine-tuning of programs.

To force the values of digital inputs or outputs when in online mode, either modify their configuration properties or use an animation table ([see page 113](#)).

Online Mode Modifications

Overview

You can modify your program, in both the Instruction List (IL) and Ladder (LD) editors, while in online mode. However, there are certain limits to the type of editing that you may do, and the instructions that you may edit, depending on whether your controller is in *RUN* or *STOP*. These limits help protect the state of the controller and the integrity of the program.

Sending Modifications in Online Mode

In IL, the modifications, when allowed, are automatically sent to the logic controller after validation of the current IL edited line. If the modification is not allowed, a message appears.

In Ladder, the modifications are not sent automatically. A **Send** button enables you to send the modifications to the logic controller in *STOP/RUN* mode. This button is only available when the program is valid. A **Rollback** button enables you to restore the initial rung.

In either case, rungs that are modified are evaluated for their viability in the context of whether the controller is in *RUN* or *STOP*. Some modifications will be accepted, while other modifications will be rejected because they would cause run time errors, or change the structure of the program memory.

The following table indicates in which cases modifications are allowed:

Operations	In <i>STOP</i> in IL	In <i>RUN</i> in IL	In <i>STOP</i> in Ladder	In <i>RUN</i> in Ladder
Event task content	editable	rejected	editable	non-editable
Master / Periodic task content	editable	editable	editable	editable
Free POU content	editable	editable	non-editable	non-editable
Rung with label	editable	rejected	non-editable	non-editable
Rung with end, jump, or calling a subroutine or label	non-editable	non-editable	non-editable	non-editable
Rung with any Grafset instruction	non-editable	non-editable	non-editable	non-editable
Add/modify label	non-editable	non-editable	non-editable	non-editable

NOTE: This table does not take into account all program structure modifications. In each rung where modifications are not allowed, the entire rung is disabled (gray foreground on the rung).

Chapter 7

Commissioning

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	Overview of the Commissioning Window	176
7.2	Managing the Connection to a Logic Controller	177
7.3	SoMachine Basic Simulator	185
7.4	Backing Up and Restoring Controller Memory	200
7.5	Downloading and Uploading Programs	202

Section 7.1

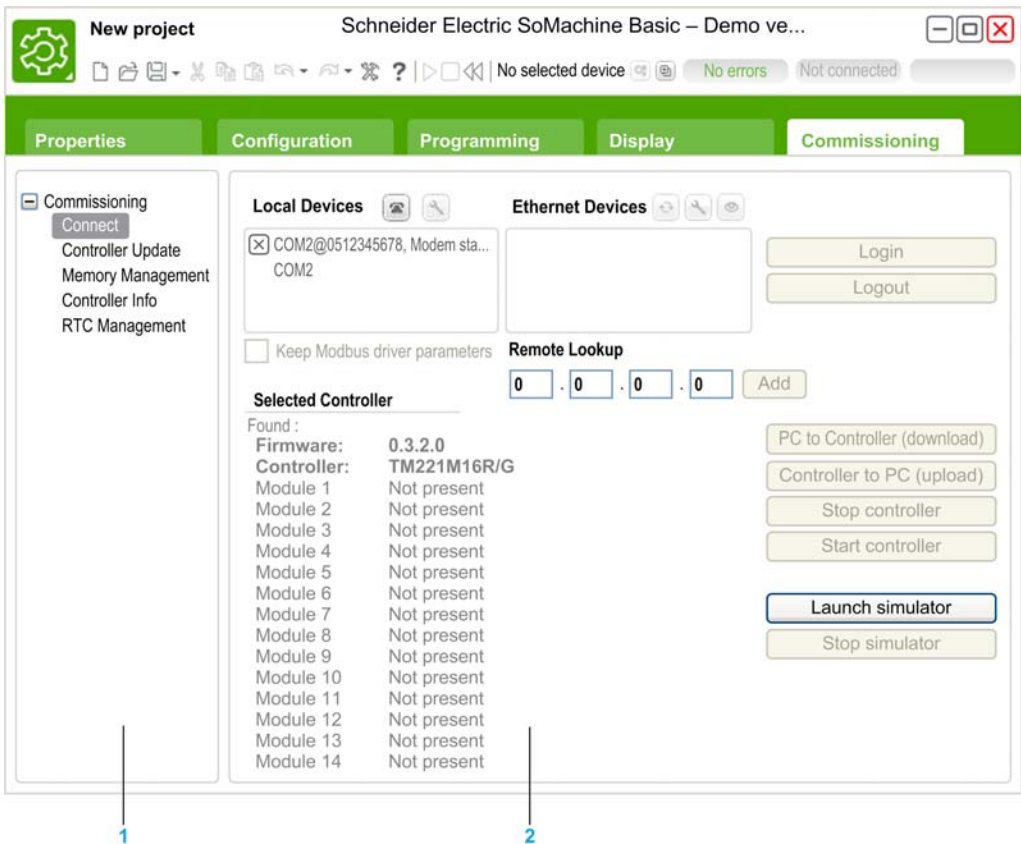
Overview of the Commissioning Window

Overview of the Commissioning Window

Introduction

The **Commissioning** window allows you:

- Login to or logout from a logic controller.
- Manage logic controller memory, for example, by performing back up and restore operations.
- Manage the Real Time Clock (RTC) of the logic controller.



- 1 The Commissioning Tree displays the available commissioning tasks.
- 2 The right-hand area allows you to perform operations of the commissioning task.

Section 7.2

Managing the Connection to a Logic Controller

What Is in This Section?

This section contains the following topics:

Topic	Page
Connecting to a Logic Controller	178
Controller Information	182
Managing the RTC	184

Connecting to a Logic Controller

Overview

Click **Connect** on the Commissioning window to manage the connection with the logic controller.

Connected Controllers

2 lists of logic controllers are displayed:

1. Local Devices

Displays all logic controllers connected to the PC:

- with the physical COM ports of the PC (COM1, for example),
- with USB cables,
- through the virtualized COM ports (by USB-to-serial converters or Bluetooth dongles),
- with modem connection that you choose to add manually.

NOTE: If a COM port is selected and the **Keep Modbus driver parameters** check box is ticked, the communication is established with the parameters defined in the Modbus driver.

2. Ethernet Devices

Displays all logic controllers that are accessible by Ethernet (on the same subnet and not under a router or any device that blocks UDP broadcasts). This list includes logic controllers that are automatically detected by SoMachine Basic as well as any controllers that you choose to add manually.

Manually Adding Ethernet Controllers

To manually add a logic controller to the **Ethernet Devices** list:

Step	Action
1	In the Remote lookup field, type the IP address of the logic controller to add, for example 12.123.134.21
2	Click Add to add the device to the Ethernet Devices list.


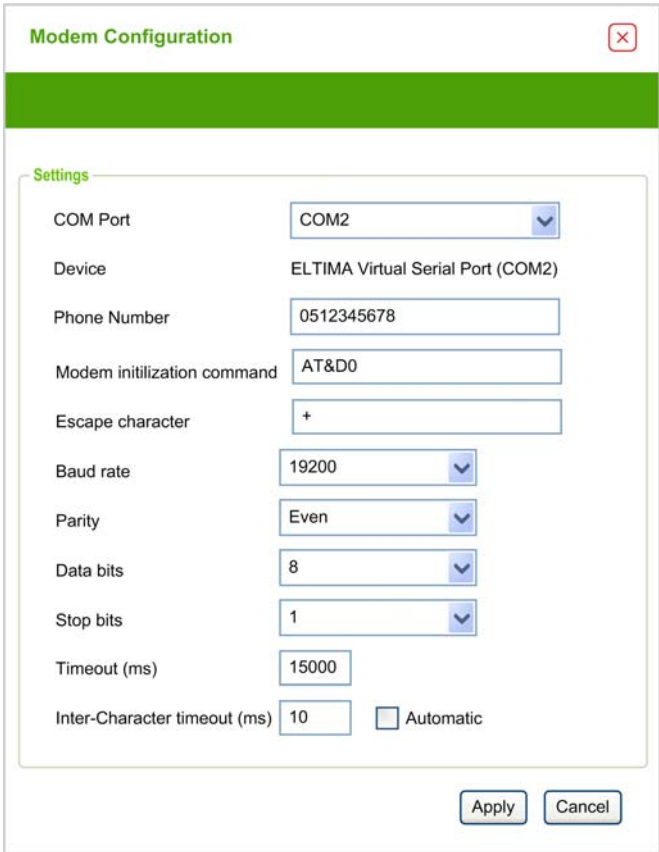
Manually Adding Modem Connection

Prerequisites for availability of modem:

- If no modem is installed on the PC, the button is disabled.
- Verify in the **Phone and Modem** option of the Windows **Control Panel** that the modem is installed and perform a test (in the **Modem** tab, click the modem to test and click **Properties** → **Diagnostics** → **Query Modem**). The response of the modem must be valid.
- If the modem is an external modem connected on a COM port, verify that the communication settings are the same in:
 - the modem advanced parameters,
 - the communication port parameters,
 - the Modbus driver parameters.

For more details on the installation and setting of modems SR2MOD02 and SR2MOD03, refer to *SR2MOD02 and SR2MOD03 Wireless Modem User Guide* (EIO00000001575.00).

To manually add a modem connection to the **Local Devices** list:

Step	Action
2	 <p>Click (Add modem configuration) button to open the Modem configuration window. Result: The Modem configuration window appears.</p>
3	<p>Select the COM port of the modem from the drop-down list:</p> 
4	<p>Configure the communication parameters. For detailed information on the modem configuration parameters, refer to the table below.</p>

Step	Action
5	<p>Click Apply.</p> <p>NOTE: This button is enabled only if all settings are correctly configured.</p> <p>Result: The modem connection is added to the Local Devices list (for example COM2@0612345678,GenericModem).</p>




This table describes each parameter of the modem configuration:

Parameter	Value	Default value	Description
Port	COMx	-	Allows you to select the COM port of the modem from the dropdown list.
Device	-	-	Displays the modem name.
Phone Number	-	-	Type the phone number of the remote modem connected to the logic controller. This text field accepts all the characters and is limited to 32 characters in total. This field must contain at least one character to be able to apply the configuration.
AT init cmd	-	AT&D0	Allows you to edit the AT initialization command of the modem. The AT initialization command is not mandatory (if the field is empty the AT string is sent).
Escape chars	-	+	Allows you to edit the escape character for the hang-up procedure.
Baud rate	1200 2400 4800 9600 19200 38400 57600 115200	19200	Allows you to select the data transmission rate (bits per second) of the modem.
Parity	None Even Odd	Even	Allows you to select the parity of the transmitted data for error detection.
Data bits	7 8	8	Allows you to select the number of data bits.
Stop bits	1 2	1	Allows you to select the number of stop bits.
Timeout (ms)	0...60000	15000	Allows you to specify the transmission timeout (in ms).

Parameter	Value	Default value	Description
Break timeout (ms)	0...10000	10	Allows you to specify the interframe timeout (in ms). If the check box Automatic is ticked, the value is automatically calculated.

Connecting to a Logic Controller

To log in to a logic controller:

Step	Action
1	Click  (Refresh Devices button) to refresh the list of connected devices.
2	Select one of the logic controllers in the Local Devices or Ethernet Devices lists. If a controller is connected by Ethernet on the same network cable than your PC, the IP address of the controller appears in the list. Selecting the IP address in the list enables  (IP Address Configuration button). Click this button to change the IP address of the controller. NOTE: If you tick the check box Write to post configuration file , the Ethernet parameters are modified in the post configuration file and kept after a power cycle.
3	If required, click  (Start Flashing LEDs button) to flash the LEDs of the selected controller to identify the controller physically by its flashing LEDs. Click this button again to stop flashing the LEDs.
4	Click Login button to log in to the selected controller. If the logic controller is password protected, you are prompted to provide the password. Type the password and click OK to connect. Result: A status bar appears showing the connection progress.
5	When connected, the protection status of the application currently stored in the logic controller appears in the Selected Controller area of the window. When the connection is successfully established, details about the logic controller appear in the Selected Controller area of the window: <ul style="list-style-type: none"> ● The firmware revision ● The logic controller reference number ● The reference numbers of all expansion modules connected to the logic controller ● The current state of the connection between SoMachine Basic and the logic controller.
6	SoMachine Basic checks whether the hardware configuration of the logic controller is compatible with the configuration of the current project. If so, the application can be downloaded to the controller. The PC to Controller (download) button is enabled and you can proceed to download the application (<i>see page 203</i>).

Controller Information

Overview

Click **Controller info** in the left-hand area of the **Commissioning** window to display the following information on the present state of the logic controller:

- **Executable RAM:** This option verifies if a valid application is stored in the random access memory (RAM) of the logic controller.
- **Protected RAM:** This option is checked if the application in the RAM of the logic controller is password protected.
- **Forced I/O:** This option is checked if 1 or more digital inputs or outputs on the logic controller are being forced to a specific value (*see page 114*).
- **Status:** The present state of the logic controller.
This information can also be obtained from within a program by testing the system word %SW6. For more information on controller states, see the *programming guide* of your logic controller.
- **Last stopped on:** The date and time that the logic controller was last stopped (STOP, HALT, and so on).
This information can also be obtained from within a program by testing the system word %SW54 through %SW57.
- **Last stopped reason:** Displays the reason for the most recent stop of the logic controller.
This information can also be obtained from within a program by testing the system word %SW58.
- **Scan time (µs):** The following scan times:
 - **Minimum** (in microseconds): Shortest scan time since the last power-on of the logic controller.
This information can also be obtained from within a program by testing the system word %SW32 (in milliseconds).
 - **Current** (in microseconds): The scan time.
This information can also be obtained from within a program by testing the system word %SW30 (in milliseconds).
 - **Maximum** (in microseconds): The longest scan time since the last power-on of the logic controller.
This information can also be obtained from within a program by testing the system word %SW31 (in milliseconds).
- **Controller time:** The following information is displayed only if the logic controller has a real time clock (RTC):
 - **Date** (DD/MM/YYYY): The present date stored in the logic controller.
This information can also be obtained from within a program by testing the system words %SW56 and %SW57.
 - **Time** (HH:MM:SS): The present time stored in the logic controller.
This information can also be obtained from within a program by testing the system words %SW54 and %SW55.

The date and time are presented in same format as specified for the PC.

-
- **Ethernet information:** The following information is displayed only if the logic controller has an embedded Ethernet connection:
 - **IP address:** The IP address of the logic controller.
This information can also be obtained from within a program by testing the system words %SW33 and %SW34.
 - **Subnet mask:** The subnet mask of the logic controller.
This information can also be obtained from within a program by testing the system words %SW35 and %SW36.
 - **Gateway address:** The gateway address of the logic controller.
This information can also be obtained from within a program by testing the system words %SW37 and %SW38.
 - **Post Configuration status for SL1:** The parameters with the activated check box are defined by the post configuration file.
 - **Post Configuration status for SL2:** The parameters with the activated check box are defined by the post configuration file.
 - **Post Configuration status for Ethernet:** The parameters with the activated check box are defined by the post configuration file.

Managing the RTC

Overview

The **RTC Management** window enables you to set the real time clock (RTC) of the logic controller. This is only possible if SoMachine Basic is connected to a logic controller that supports an RTC.

Updating the RTC

Step	Action
1	Select the RTC Management option in the left-hand area of the Commissioning window.
2	If in online mode, the Current controller time is displayed. Choose the mode for setting the logic controller time: <ul style="list-style-type: none">● Manual : This mode displays the date and time and lets you manually choose what date and time to set in the logic controller.● Automatic : This mode sets the time in the logic controller to the current time of the PC on which SoMachine Basic is installed.
3	Click Apply .

Section 7.3

SoMachine Basic Simulator

What Is in This Section?

This section contains the following topics:

Topic	Page
Overview of the SoMachine Basic Simulator	186
SoMachine Basic Simulator I/O Manager Window	188
SoMachine Basic Simulator Time Management Window	190
Modifying Values Using SoMachine Basic Simulator	193
How to Use the SoMachine Basic Simulator	198
Launching Simulation in Vijeo-Designer	199

Overview of the SoMachine Basic Simulator

Introduction

SoMachine Basic simulator allows you:

- to simulate a connection between the PC, the logic controller, and any expansion modules.
- to run and test a program without a logic controller and expansion modules, connected to the PC physically.

The philosophy of operation is such that the simulator replicates the behavior of the controller and is, in fact, a virtual controller that you connect to with SoMachine Basic. Once you launch the simulator, you can connect, run, stop and other associated actions that you would normally accomplish while connected to a physical controller.

Accessing the SoMachine Basic Simulator

You can launch the simulator by any of the following methods:

- Click **Launch simulator** button in the commissioning task area.
- Press CTRL+B in the **Commissioning** window.

- Click  (launch simulator button) in the SoMachine Basic tool bar.

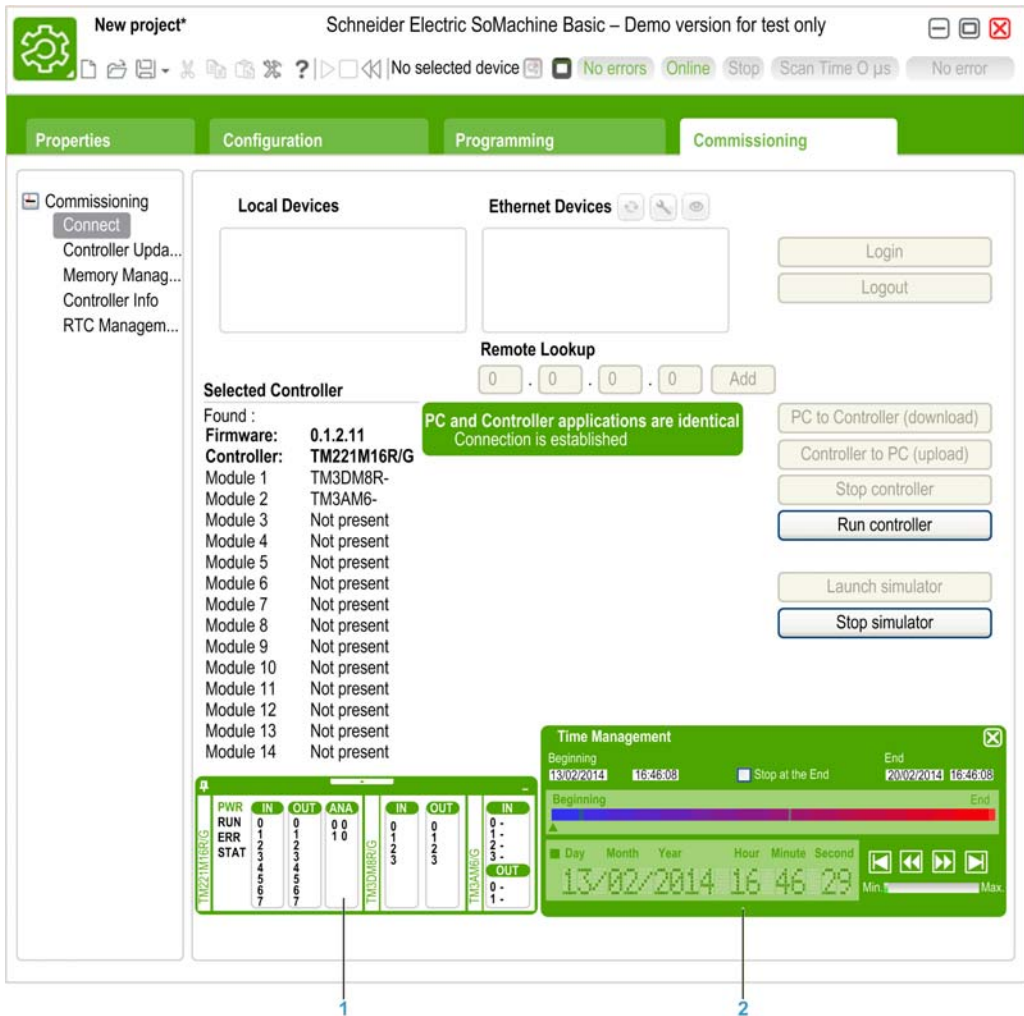
Before launching the simulator, ensure that the program is valid. Otherwise, the simulator launch is interrupted with a compilation error detected message that appears on the screen.

SoMachine Basic Simulator Windows

SoMachine Basic simulator has the following 2 windows:

- **Simulator time management window**
Lets you control the RTC of the controller in order to simulate the passage of time and its effect on the logical constructs affected by the RTC.
- **Simulator I/O manager window**
Lets you manage the state of inputs/outputs of the controller and the expansion modules.

After the connection between the PC and the virtual logic controller has been successfully established (see How to Use the SoMachine Basic Simulator ([see page 198](#))), SoMachine Basic simulator windows appear on the screen:



- 1 Simulator I/O manager window ([see page 188](#))
- 2 Simulator Time Management window ([see page 190](#))

SoMachine Basic Simulator I/O Manager Window

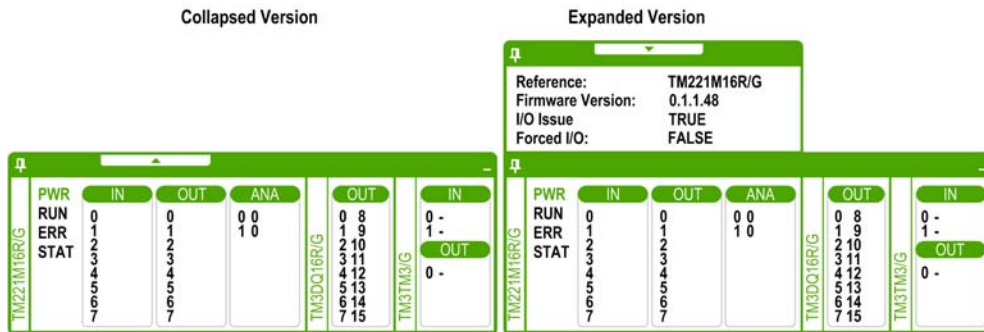
Overview

Simulator I/O manager window has the following components:

- LED status:
To monitor the LED status of a simulated controller.
- Input/output status:
To control the inputs and outputs when the program is running.

Simulator I/O Manager Window

This graphic shows the simulator I/O manager window:



Click the pin symbol on the left-top of this window to pin or unpin the window to the foreground.

Click the minimize symbol on the right-top of this window to minimize the window in taskbar.

LED Status

The PWR, RUN, ERR, and STAT LEDs are simulated in the SoMachine Basic simulator I/O manager window as they would appear on a connected base controller.

The following are the LED states displayed in the simulator I/O manager window of a simulated logic controller:

LED	Status Information
PWR	Indicates whether the simulated logic controller is powered up or not.
RUN	Indicates the RUN state of the simulated logic controller.
ERR	Indicates the ERR state of the simulated logic controller.
STAT	The operation of the STAT LED is defined by the user logic.

Input/Output Status

Simulator I/O manager window lets you monitor and control the I/Os of a controller and expansion module when a program is running.

The inputs and outputs are displayed in a list of numbers. This list depends on the I/Os of the selected controller and expansion module. For example, if your controller has n digital inputs, the number list will display number starting from $0 \dots (n-1)$, where each number corresponds to the digital input at the corresponding input channel.

For a controller, the I/Os displayed are:

- **IN**: Digital inputs.
- **OUT**: Digital outputs.
- **ANA**: Analog inputs.

For an expansion module, the I/Os displayed are:

- **IN**: Digital/analog inputs.
- **OUT**: Digital/analog outputs.

NOTE: The analog I/Os are displayed with their current values on right-hand side of the analog input number.

Digital I/O status is identified by the text color of the I/O numbers:

- Green: I/O is set to 1.
- Black: I/O is set to 0.

Analog I/O status is identified by the value:

- - (hyphen): I/O is not configured.
- Number: Current value of the I/O.

SoMachine Basic Simulator Time Management Window

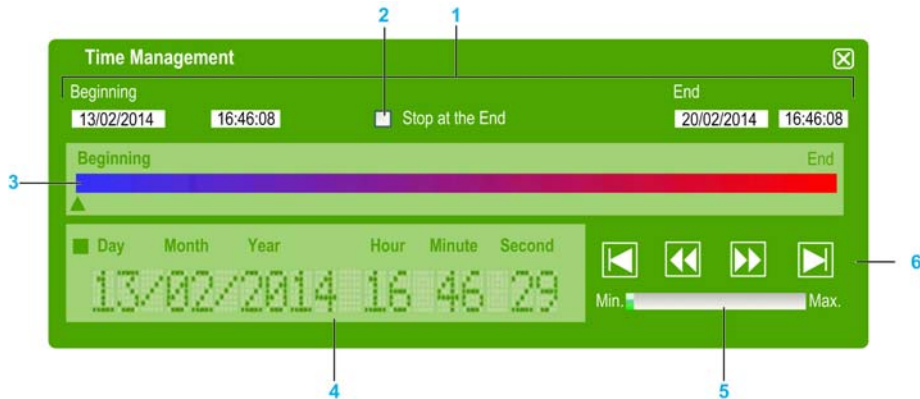
Overview

Simulator **Time Management** window has the following components:

- Date / Time simulation range for the execution of the program in the simulator:
 - **Beginning** Date and Time
 - **End** Date and Time
 - **Stop at the End** check box (stop the execution of the program when the **End** Date and Time are reached)
- Time control scroll bar:
 - To move the simulation of the passage of time manually forward or back
- Date and Time display:
 - Date and Time of the simulated RTC of the simulator
- Control buttons:
 - To reset, jump back, jump forward, or end the time management associated with the RTC
- Increment bar:
 - To fix the rate of the simulated passage of time relative to real time

Simulator Time Management Window

This graphics depicts the simulator **Time Management** window:



- 1 Date / Time simulation range (Beginning – End)
- 2 Stop at the end (of Date / Time range) check box
- 3 Time control scroll bar
- 4 RTC date and time
- 5 Increment bar
- 6 Elapsed time control buttons

Simulator Date / Time Simulation Range

The simulation range allows you to establish and control the RTC of the simulator. The RTC is set with the **Beginning** date and time fields when you set the simulator into a RUN state. The **End** date and time fields establishes the end of your simulation. If you check the **Stop at the End** check box, the simulator enters a STOP state at the expiration of the simulation range. Otherwise, the simulator will continue to run, as will the RTC, until you manually stop the simulator with SoMachine Basic.

Time Control Scroll Bar

The time control scroll bar allows you to manually manipulate the date and time you have established simulation range. Click and hold the right mouse button while pointing at the arrow below the bar and moving the mouse to the right advances the time and date of the RTC. Doing the same and moving the mouse to the left reverse the time and date of the RTC.

RTC Date and Time



The RTC date and time zone displays the value of the RTC as it relates to the ongoing simulation. The initial time of the RTC is established by the **Beginning** date and time when you place the simulator in a RUN state. Thereafter, the display is updated with the ongoing clocking of the RTC in the simulator. You can alter the RTC either with the time control scroll bar or with the Time elapse speed control buttons.



Increment Bar

The increment bar allows you to establish a relative increment for Jumping forward or back the RTC value when using the elapsed time control buttons. By clicking the bar you can set various increments that are relative to the simulation range you have established.

Elapsed Time Control Buttons

You can use the control buttons to effect the RTC value, and therefore manipulate its affect on your program running in the simulator as follows:

Graphic Element	Command	Description
	Initialize	Allows you to reset the date and time back to that which is set in the Beginning time/date field.
	Jump Forward	Allows you to move forward the time and date from its current value in increments established by the Increment bar.

Graphic Element	Command	Description
	Jump Back	Allows you to reverse the time and date from its current value in increments established by the Increment bar.
	End	Allows you to jump the date and time to that which is set in the End time/date field.

Modifying Values Using SoMachine Basic Simulator

Overview

When in online mode, SoMachine Basic simulator I/O manager window allows you to:

- Modify values of the inputs.
- Trace the outputs.

Modifying Values of Digital Inputs

Follow these steps to modify the digital input value, using single-click operation:

Step	Action
1	Click the digital input number in the simulator I/O manager window to change the discrete input value. Result: Text color of the input number changes. Digital input values are identified by the text color: <ul style="list-style-type: none"> • Green: I/O is set to 1. • Black: I/O is set to 0.
2	Click again on the same input number to toggle the value.

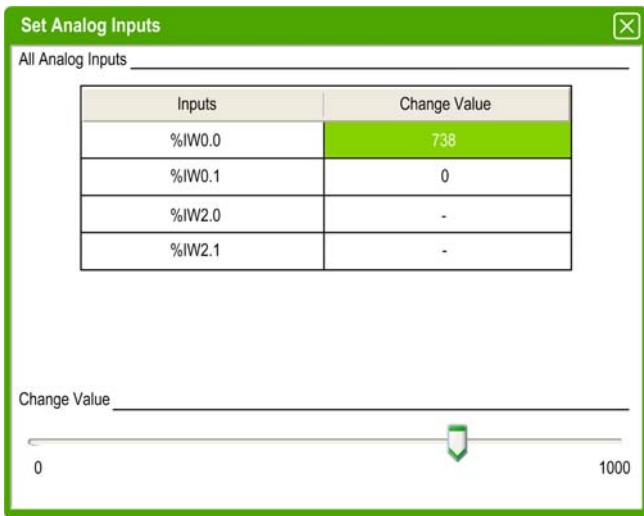
Follow these steps for bulk operation of modifying digital input values together:

Step	Action																
1	Double-click the digital input number in the simulator I/O manager window, Result: Set Discrete Inputs window listing all digital inputs, appears on the screen: <div data-bbox="304 850 853 1393" style="border: 1px solid green; padding: 10px; margin: 10px 0;"> <p style="margin: 0;">Set Discrete Inputs</p> <p style="margin: 0;">All Discrete Inputs</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 70%;">Inputs</th> <th style="width: 30%;">Set to I/O</th> </tr> </thead> <tbody> <tr> <td>%I0.0</td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> <tr style="background-color: #90EE90;"> <td>%I0.1</td> <td style="text-align: center;"><input checked="" type="checkbox"/></td> </tr> <tr> <td>%I0.2</td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> <tr> <td>%I0.3</td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> <tr> <td>%I0.4</td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> <tr> <td>%I0.5</td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> <tr> <td>%I0.6</td> <td style="text-align: center;"><input type="checkbox"/></td> </tr> </tbody> </table> <p style="margin: 10px 0;">Operation</p> <div style="display: flex; justify-content: space-around; margin: 5px 0;"> Set all to 0 Set all to 1 </div> <div style="display: flex; justify-content: space-between; margin-top: 10px;"> Help Cancel OK </div> </div>	Inputs	Set to I/O	%I0.0	<input checked="" type="checkbox"/>	%I0.1	<input checked="" type="checkbox"/>	%I0.2	<input type="checkbox"/>	%I0.3	<input type="checkbox"/>	%I0.4	<input type="checkbox"/>	%I0.5	<input type="checkbox"/>	%I0.6	<input type="checkbox"/>
Inputs	Set to I/O																
%I0.0	<input checked="" type="checkbox"/>																
%I0.1	<input checked="" type="checkbox"/>																
%I0.2	<input type="checkbox"/>																
%I0.3	<input type="checkbox"/>																
%I0.4	<input type="checkbox"/>																
%I0.5	<input type="checkbox"/>																
%I0.6	<input type="checkbox"/>																

Step	Action
2	In the Operation area of the Set Discrete Inputs window, click: <ul style="list-style-type: none"> ● Set all to 0: To set the value of all inputs to 0. ● Set all to 1: To set the value of all inputs to 1. <p>Result: If the checkbox is selected, input value is set to 1. If not selected, input value is set 0.</p>
3	Alternatively, in the All Discrete Inputs area of the Set Discrete Inputs window, click the checkbox corresponding to the input to modify the values individually.
4	Click OK to save the changes and exit from the Set Discrete Inputs window.

Modifying I/O Values of Analog Inputs


Follow these steps for modifying analog input values:

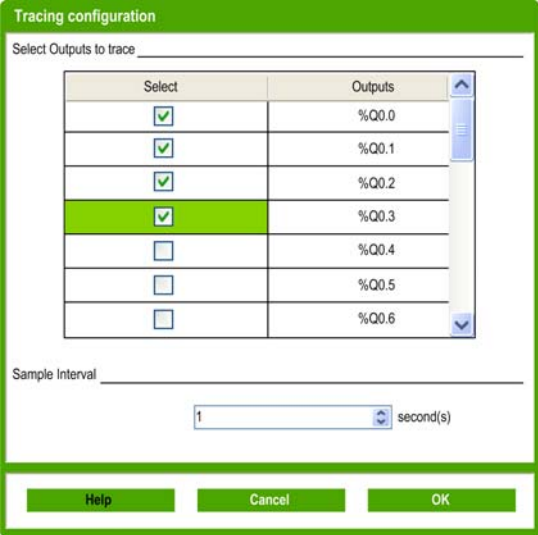
Step	Action
1	Double-click the analog input number in the simulator I/O manager window, Result: Set Analog Inputs window listing all analog inputs, appears on the screen:
	
2	In the All Analog Inputs area of the Set Analog Inputs window, double-click the value field in the Change Value column corresponding to the input to be modified.
3	Enter the value in the range 0...1023 and press ENTER.
4	Alternatively, in the Set Analog Inputs window, select an input from the Inputs list and move the slider in the Change Value area to adjust the input value between 0...1023. When you move the slider from left to right, the value increases and vice versa.
5	Click OK to save the changes and exit from the Set Analog Inputs window.

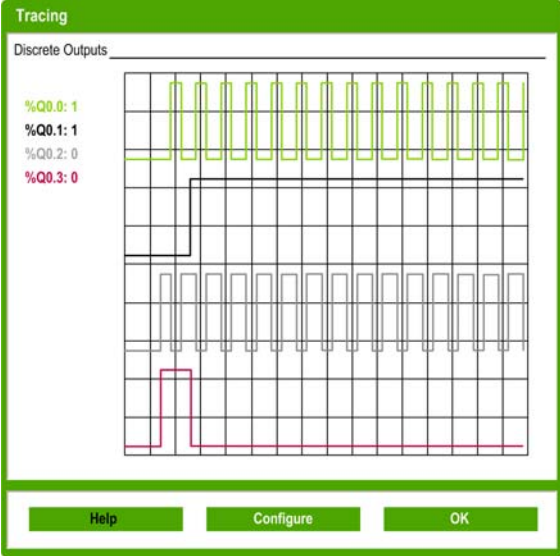
Tracing the Outputs

Output values depend on the program; therefore you cannot modify the values but SoMachine Basic simulator offers you to trace the digital and analog outputs.

Follow these steps for modifying analog input values:

Step	Action
1	<p>Double-click the output number in the simulator I/O manager window, Result: Tracing window appears on the screen.</p> 

Step	Action
2	<p>Click Configure button to select the outputs to trace. Result: Tracing Configuration window appears on the screen.</p> 
3	<p>In the Select checkbox column, click the checkboxes corresponding to the outputs to trace.</p>
4	<p>Select the Sample Interval from the drop-down menu to set the sample time interval for output tracing:</p> <ul style="list-style-type: none"> ● 1 second ● 5 seconds ● 10 seconds ● 20 seconds

Step	Action
5	<p>Click OK to save and exit from the Tracing Configuration window.</p> <p>Result: Selected outputs are added to the Tracing window that displays tracing of the outputs with simulated values:</p>  <p>The screenshot shows a window titled "Tracing" with a green border. Inside, there's a section "Discrete Outputs" with a grid. Four traces are visible: %Q0.0: 1 (green), %Q0.1: 1 (green), %Q0.2: 0 (black), and %Q0.3: 0 (pink). The traces show digital signals over time. At the bottom are buttons for Help, Configure, and OK.</p>
6	Click OK to exit from the Tracing window.

How to Use the SoMachine Basic Simulator

Procedure

Follow these steps to run the SoMachine Basic Simulator to test your program:

Step	Action
1	Ensure that you have a valid program by checking the status message in the status area (for more information, refer to Status Area (see page 47)). The program status should be No errors . You can also run SoMachine Basic simulator when the program status is Advice .
2	Launch the simulator (refer to Accessing the Simulator (see page 186)).
3	Run the controller. In the Commissioning window, select Connect in the commissioning tree and then click Run controller button in the commissioning task area.
4	Command your program using the simulator main window (refer to Control Buttons (see page 191)).
5	Check the LED status in the simulator main window (refer to LED Display (see page 189)).
6	Check the status of the inputs/outputs in the simulator I/O manager window (refer to Input/Output Status (see page 189)).
7	Check the LED status in the simulator I/O manager window (refer to LED Status (see page 188)).
8	Modify the I/O values as required (refer to Modifying Values Using the Simulator (see page 193)).
9	Trace the outputs as required (refer to Tracing the Outputs (see page 195)).
10	Stop the controller. In the Commissioning window, select Connect in the commissioning tree and then click Stop controller button in the commissioning task area.
11	Stop the simulator. In the Commissioning window, select Connect in the commissioning tree and then click Stop controller button in the commissioning task area or press CTRL+W to exit from the simulator.

Launching Simulation in Vijeo-Designer

Procedure

Prior to launching the HMI simulation in Vijeo-Designer, first start the logic controller simulator in SoMachine Basic (*see page 185*).

Follow these steps to launch the simulation in Vijeo-Designer:

Step	Action
1	Start Vijeo-Designer.
2	Open the Vijeo-Designer project that contains the symbols from a SoMachine Basic project. NOTE: If the Vijeo-Designer project does not exist, create a project in Vijeo-Designer and share the symbols with the SoMachine Basic project. For more information, refer to Sharing Symbols Between a SoMachine Basic Project and a Vijeo-Designer Project (<i>see page 127</i>).
3	Click the Project tab in the Navigator window, right-click the equipment node under the IO Manager node, and select Configuration . Result: The Equipment Configuration window opens.
4	Enter the IP Address and click OK . NOTE: The IP address must be a local host address or local address of your PC. For example, 127.0.0.1
5	Start Device Simulation Tool .
6	Click the Variables tab and select the check boxes of the variables to include in the simulation. NOTE: If the View All icon is selected, all variables selected in the Variables tab are displayed in the Simulation tab.
7	Click the Simulation tab.
8	Select a variable, select an operation for the variable, and then select the Active check box. NOTE: Only one simulation operation can be applied to any given variable at a time.
9	Define the parameters of the variable simulation operation.
10	Click the Simulation icon to start the simulation.
11	Change the variable values as required during the simulation: <ul style="list-style-type: none"> ● For a slider operation, you can change the value by moving the slider, moving the wheel on your mouse, or typing the arrow keys on the keyboard. ● For a toggle operation, click Set or Reset to write the corresponding string to the variable.
12	Click the Simulation icon again to stop the simulation.
13	Press CTRL+Z to exit the Device Simulation Tool .

Section 7.4

Backing Up and Restoring Controller Memory

Backing Up and Restoring Controller Memory

Overview

SoMachine Basic provides you the ability to back up or restore the controller memory. You can back up or restore only memory bits and memory words through controller memory management. Other objects (for example, timers, counters, and so on) are managed in application download and upload.

Back up and restore options are available only in online mode.

Backing Up to a PC


Follow these steps to backup controller memory to a PC:

Step	Action
1	In the Commissioning tab, select Memory Management .
2	Under Action , choose Backup from Controller .
3	Under Destination file , choose PC . Click the browse button, navigate to the folder in which to write the backup file, and type the name of the backup file (* . csv).
4	Select Back up memory variables to include the logic controller memory in the backup. Specify the First Memory Bit , Last Memory Bit , First Memory Word , and Last Memory Word to be included in the backup.
5	Click Backup from Controller button to begin the backup operation. A report window appears displaying a list of information or detected error messages about the memory backup operation. The last line of this list displays Memory backup succeeded if the operation was successful. If the memory backup operation is unsuccessful, a message appears in the last line of the report window and the incomplete files (* . csv) are automatically deleted.

NOTE: You can initiate a backup while the controller is in a **[RUN]** state. However, depending on the amount of memory variables you specify to be included in the backup, it may be that the backup can not be accomplished between logic scans. As a consequence, the backup would not necessarily be coherent such that the value of memory variables could be modified from one scan to another. If you wish to have a consistent set of values for the variables, you may need to first put the controller into a **[STOP]** state.

Restoring from a PC

Follow these steps to restore controller memory from a PC:

Step	Action
1	<p>Ensure that the controller is in STOPPED state. If the controller is in RUN state, perform any of the following actions to stop the controller:</p> <ul style="list-style-type: none"> Click stop controller icon  in the toolbar at the top of SoMachine Basic window. In the Commissioning window, select Connet in the commissioning tree and then click Stop Controller button in the commissioning task area.
2	In the Commissioning tab, select Memory Management .
3	In the Action list, choose Restore to Controller .
4	Under Source file choose PC to restore the controller memory from a file stored on the PC. Click the browse button, navigate to the folder containing the file, and select the previously backed up file (*.csv).
5	<p>Click Restore to Controller to begin the restore operation. A report window appears displaying a list of information or detected error messages about the memory restore operation. The last line of this list displays Memory restoration succeeded if the operation was successful. If the memory backup operation is unsuccessful, a message appears in the last line of the report window.</p>

If there is a power outage or communication interruption during the restore of the application data, your machine may become inoperative. If a communication interruption or a power outage occurs, reattempt the restore.

NOTICE

INOPERABLE EQUIPMENT

- Do not interrupt the restore of the application data once the restore has begun.
- Do not place your machine into service until the restore has completed successfully.

Failure to follow these instructions can result in equipment damage.

Section 7.5

Downloading and Uploading Programs

What Is in This Section?

This section contains the following topics:

Topic	Page
Downloading and Uploading Applications	203
Controller Updates	205

Downloading and Uploading Applications

Downloading an Application

Follow these steps to download an application currently stored in SoMachine Basic to the logic controller:

Step	Action
1	Click Connect in the commissioning tree of the Commissioning window.
2	Select one of the logic controllers in the Local Devices or Ethernet Devices lists.
3	Click Login button to log in to the selected controller. If the logic controller is by password protected, type the password and click OK to connect.
4	Click PC to Controller (download) . If the PC to Controller (download) button is not available, confirm whether: <ul style="list-style-type: none"> • The application stored in the logic controller is identical to the current SoMachine Basic application. • The hardware configuration of the logic controller system is not compatible with the current configuration in the SoMachine Basic application.
5	If the application has been configured to Start in Run , a hazard message is displayed and prompts you to confirm that the application has been so configured. Click OK to confirm the download of the application or click Cancel and modify the configuration.
6	Click OK to continue the transfer and overwrite the current logic controller application. Result: A status bar appears showing the connection status.
7	To run the application you have downloaded, click Run controller and click OK to confirm the action. If a message appears informing you that the operating mode cannot be changed, click Close and check whether the RUN/STOP switch on the logic controller is in the STOP position or, alternatively, if the RUN/STOP input is configured, that it is not additionally preventing the controller from passing into RUN. Otherwise, refer to the <i>Hardware Guide</i> of your logic controller for details.

Uploading an Application

Follow these steps to upload an application currently stored in the logic controller to SoMachine Basic:

Step	Action
1	Click Connect in the commissioning tree of the Commissioning window.
2	Select one of the logic controllers in the Local Devices or Ethernet Devices lists.
3	Click Login button to log in to the selected controller. If the logic controller is by password protected, type the password and click OK to connect.
4	Click Controller to PC (upload) . If the Controller to PC (upload) button is not available, confirm whether: <ul style="list-style-type: none">• The application stored in the logic controller is identical to the current SoMachine Basic application.• The hardware configuration of the logic controller system is not compatible with the current configuration in the SoMachine Basic application.
5	Click OK to confirm upload from the logic controller. Result: A status bar appears showing the connection status. When the transfer completes, the application is uploaded from the logic controller into SoMachine Basic.

Controller Updates

Overview

You can download firmware updates to the logic controller either directly from SoMachine Basic or using an SD card.

Sending a Firmware Update to the Logic Controller

Performing a firmware change will delete the current application program in the device, including the Boot Application in Flash memory.

NOTICE

LOSS OF APPLICATION DATA

- Perform a backup of the application program to the hard disk of the PC before attempting a firmware change.
- Restore the application program to the device after a successful firmware change.

Failure to follow these instructions can result in equipment damage.

Follow these steps to send firmware updates to the logic controller:

Step	Action
1	Log in (<i>see page 178</i>) to the logic controller. If the logic controller is password protected, type the password and click OK to connect.
2	Click Commissioning → Controller Update .
3	Click the Browse button next to the Select Firmware File box and select the firmware file (*.mfw) to download to the controller.
4	Click OK . Result: The firmware download begins. Status or detected error messages are displayed in the Details box.

If there is a power outage or communication interruption during the transfer of the application program or a firmware change, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer.

NOTICE

INOPERABLE EQUIPMENT

- Do not interrupt the transfer of the application program or a firmware change once the transfer has begun.
- Do not place the device into service until the transfer has completed successfully.

Failure to follow these instructions can result in equipment damage.

Chapter 8

Saving Projects and Closing SoMachine Basic

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Saving a Project	208
Saving a Project As a Template	209
Closing SoMachine Basic	210


Saving a Project

Overview


SoMachine Basic projects can be saved as files to the local PC. This file has the extension * .smbp and contains:

- The source code of the program contained on the **Programming** tab
- The current hardware configuration contained on the **Configuration** tab
- Settings and preferences set in the SoMachine Basic project.

Saving the Project

Step	Action
1	Click Save  on the toolbar, or press Ctrl-S
2	If this is the first time you have saved the project, browse and select the folder in which to store the project file.
3	Type the name of the project file and click Save .

Saving the Project with a Different Name


Step	Action
1	Click the menu arrow next to the Save button  on the toolbar and choose Save as .
2	Browse and select the folder in which to store the project file.
3	Type the new name of the project file and click Save .

Saving a Project As a Template

Overview

SoMachine Basic projects can be saved as templates. The project is then listed on the **Templates** tab of the Start page (*see page 38*). You can then use the project as the starting point for new projects.

Saving a Project as a Template

Step	Action
1	Click the menu arrow next to the Save button  on the toolbar and choose Save as template .
2	If it is not already selected, browse to the Examples folder in your SoMachine Basic installation folder.
3	Type the name of the project.
4	Choose Sample Project Files (*.smbe) as the file Type and click Save .

Closing SoMachine Basic

Overview

To exit SoMachine Basic, click the **Close** button in the top right-hand corner of the SoMachine Basic window.

You can also click the **Exit** button on the **Start page** window.

Appendices



Appendix A

SoMachine Basic Keyboard Shortcuts

SoMachine Basic Keyboard Shortcuts

Keyboard Shortcuts List

Modifier	Key	Command	View	Condition
CTRL	C	Copy	Textbox	–
CTRL	V	Paste	Textbox	–
CTRL	X	Cut	Textbox	–
ALT	Left	Go to previous tab	All	–
ALT	Right	Go to following tab	All	–
	F1	Show help	All	–
SHIFT	F1	Show contextual help	All	–
ALT	F4	Exit SoMachine Basic	All	–
CTRL	B	Run simulator	All	–
CTRL	G	Login	All	–
CTRL	H	Logout	All	–
CTRL	L	Stop controller	All	–
CTRL	M	Run controller	All	–
CTRL	N	New project	All	–
CTRL	O	Open project	All	–
CTRL	P	Print project report	All	–
CTRL	Q	Exit SoMachine Basic	All	–
CTRL	S	Save project	All	–
CTRL	W	Stop simulator	All	–
CTRL	J	Download	Commissioning	–
CTRL	K	Upload	Commissioning	–
	ALT	Show Ladder shortcuts	Programming	–
	Del	Delete	Programming	items are selected
CTRL	D	Convert all rungs in program to Ladder	Programming	–
CTRL+ALT	D	Convert all rungs in program to IL	Programming	–

Modifier	Key	Command	View	Condition
CTRL	F	Search	Programming	–
CTRL	I	Insert a new rung before the selected rung	Programming	–
CTRL	Y	Redo	Programming	–
CTRL	Z	Undo	Programming	–
CTRL	Arrow key	Draw line	Ladder rung	Drawing tool selected
CTRL	Arrow key	Erase line	Ladder rung	Erasing tool selected
CTRL	Arrow key	Select/unselect next ladder cell (cell by cell)	Ladder rung	Selection tool selected
SHIFT	Arrow key	Select/unselect next ladder cells (select by area)	Ladder rung	Selection tool selected
	ESC	Reset pointer to selection tool	Ladder rung	Selected tool is not draw wire or erase wire, no items are being dragged, no popup is opened
	ESC	Cancel the pending line	Ladder rung	Drawing in progress
	ESC	Cancel the erasing line	Ladder rung	Erasing in progress
	ESC	Cancel move selected item(s) (restore initial position)	Ladder rung	Ladder items are being dragged
	ESC	Close suggestion's list	Ladder rung	A suggestions list is opened (like the available descriptors for a contact)
	ESC	Close menu item of ladder toolbar	Ladder rung	A menu of the ladder toolbar is opened (like function blocks)
	ENTER	Start/stop moving ladder elements	Ladder rung	At least one selected cell
	Arrow key	Move floating cell	Ladder rung	Moving cell started
	Arrow key	Change current cell	Ladder rung	By default
	F5	Open contact	Ladder rung	Asian set 1 ladder toolbar
	F6	Open branch	Ladder rung	Asian set 1 ladder toolbar
SHIFT	F5	Close contact	Ladder rung	Asian set 1 ladder toolbar
SHIFT	F6	Close branch	Ladder rung	Asian set 1 ladder toolbar
	F7	Coil	Ladder rung	Asian set 1 ladder toolbar
CTRL	F7	Negated coil	Ladder rung	Asian set 1 ladder toolbar
CTRL	F5	Set coil	Ladder rung	Asian set 1 ladder toolbar
CTRL	F6	Reset coil	Ladder rung	Asian set 1 ladder toolbar
	F8	Application instruction	Ladder rung	Asian set 1 ladder toolbar
	F9	Draw horizontal line	Ladder rung	Asian set 1 ladder toolbar

Modifier	Key	Command	View	Condition
	F10	Draw vertical line	Ladder rung	Asian set 1 ladder toolbar
CTRL	F9	Delete horizontal line	Ladder rung	Asian set 1 ladder toolbar
CTRL	F10	Delete vertical line	Ladder rung	Asian set 1 ladder toolbar
SHIFT	F7	Rising pulse open contact	Ladder rung	Asian set 1 ladder toolbar
SHIFT	F8	Falling pulse open contact	Ladder rung	Asian set 1 ladder toolbar
ALT	F7	Rising pulse open branch	Ladder rung	Asian set 1 ladder toolbar
ALT	F8	Falling pulse open branch	Ladder rung	Asian set 1 ladder toolbar
CTRL+SHIFT	O	Comparison block	Ladder rung	Asian set 1 ladder toolbar
	X	XOR blocks	Ladder rung	Asian set 1 ladder toolbar
	F	Function blocks	Ladder rung	Asian set 1 ladder toolbar
	A	Activate step	Ladder rung	Asian set 1 ladder toolbar
	D	Deactivate step	Ladder rung	Asian set 1 ladder toolbar
CTRL+ALT	F10	Reverse operation results	Ladder rung	Asian set 1 ladder toolbar
	O	Other Ladder items	Ladder rung	Asian set 1 ladder toolbar
ALT	F10	Free-drawn line	Ladder rung	Asian set 1 ladder toolbar
ALT	F9	Delete free-drawn line	Ladder rung	Asian set 1 ladder toolbar
	C	New contact	Ladder rung	Asian set 2 ladder toolbar
	/	New close contact	Ladder rung	Asian set 2 ladder toolbar
	W	New contact OR	Ladder rung	Asian set 2 ladder toolbar
	X	New close contact OR	Ladder rung	Asian set 2 ladder toolbar
CTRL+SHIFT	F4	Rising edge	Ladder rung	Asian set 2 ladder toolbar
CTRL+SHIFT	F5	Falling edge	Ladder rung	Asian set 2 ladder toolbar
CTRL+SHIFT	O	Comparison block	Ladder rung	Asian set 2 ladder toolbar
ALT	X	XOR blocks	Ladder rung	Asian set 2 ladder toolbar
	F10	New vertical line	Ladder rung	Asian set 2 ladder toolbar
ALT	L	New horizontal line	Ladder rung	Asian set 2 ladder toolbar
	O	New coil	Ladder rung	Asian set 2 ladder toolbar
	Q	New close coil	Ladder rung	Asian set 2 ladder toolbar
CTRL+SHIFT	F9	Set coil	Ladder rung	Asian set 2 ladder toolbar
CTRL+SHIFT	F9	Reset coil	Ladder rung	Asian set 2 ladder toolbar
	A	Activate step	Ladder rung	Asian set 2 ladder toolbar
	D	Deactivate step	Ladder rung	Asian set 2 ladder toolbar
	I	New instruction	Ladder rung	Asian set 2 ladder toolbar
	F	New function block	Ladder rung	Asian set 2 ladder toolbar

Modifier	Key	Command	View	Condition
ALT	O	Other Ladder items	Ladder rung	Asian set 2 ladder toolbar
	F2	Deactivate branching mode	Ladder rung	European or American ladder toolbar
SHIFT	F2	Activate branching mode	Ladder rung	European or American ladder toolbar
SHIFT	F3	Normally open contact	Ladder rung	European ladder toolbar
SHIFT	F4	Normally close contact	Ladder rung	European ladder toolbar
CTRL+SHIFT	F4	Rising edge	Ladder rung	European ladder toolbar
CTRL+SHIFT	F5	Falling edge	Ladder rung	European ladder toolbar
CTRL+SHIFT	O	Comparison block	Ladder rung	European ladder toolbar
	X	XOR blocks	Ladder rung	European ladder toolbar
SHIFT	F7	Assignment	Ladder rung	European ladder toolbar
CTRL+SHIFT	F9	Negated coil	Ladder rung	European ladder toolbar
	F9	Set coil	Ladder rung	European ladder toolbar
SHIFT	F9	Reset coil	Ladder rung	European ladder toolbar
	A	Activate step	Ladder rung	European ladder toolbar
	D	Deactivate step	Ladder rung	European ladder toolbar
SHIFT	F5	Function block	Ladder rung	European ladder toolbar
CTRL+SHIFT	F6	Operation block	Ladder rung	European ladder toolbar
	F3	Line	Ladder rung	European ladder toolbar
	F3	Draw wire line	Ladder rung	European ladder toolbar
	F4	Erase wire line	Ladder rung	European ladder toolbar
	O	Other Ladder items	Ladder rung	European ladder toolbar
SHIFT	F2	Activate branching mode	Ladder rung	SoMachine ladder toolbar
	F2	Deactivate branching mode	Ladder rung	SoMachine ladder toolbar
	F3	Draw wire line	Ladder rung	SoMachine ladder toolbar
SHIFT	F3	Erase wire line	Ladder rung	SoMachine ladder toolbar
	F4	Normal contact	Ladder rung	SoMachine ladder toolbar
SHIFT	F4	Negated contact	Ladder rung	SoMachine ladder toolbar
CTRL	F9	Coil	Ladder rung	SoMachine ladder toolbar
CTRL+SHIFT	F9	Negative coil	Ladder rung	SoMachine ladder toolbar
	F9	Set Coil	Ladder rung	SoMachine ladder toolbar
SHIFT	F9	Reset Coil	Ladder rung	SoMachine ladder toolbar
CTRL+SHIFT	F4	Rising edge	Ladder rung	SoMachine ladder toolbar
CTRL+SHIFT	F5	Falling edge	Ladder rung	SoMachine ladder toolbar

Modifier	Key	Command	View	Condition
CTRL+SHIFT	{6, 7, 8, 9}	Operation Block	Ladder rung	SoMachine ladder toolbar
CTRL+SHIFT	{O, P, Q, R, S, T}	Comparison Block	Ladder rung	SoMachine ladder toolbar
	X or ALT+X	XOR blocks	Ladder rung	SoMachine ladder toolbar
	O or ALT+O	Other Ladder items	Ladder rung	SoMachine ladder toolbar
	A or ALT+A	Activate step	Ladder rung	SoMachine ladder toolbar
	D or ALT+D	Deactivate step	Ladder rung	SoMachine ladder toolbar



!

%S

According to the IEC standard, %S represents a system bit.

%SW

According to the IEC standard, %SW represents a system word.

A

animation table

A software table that displays the real-time values of objects such as input bits and memory words. When SoMachine Basic is connected to a logic controller, the values of certain object types in animation tables can be forced to specific values. Animation tables are saved as part of SoMachine Basic applications.

application

A program including configuration data, symbols, and documentation.

C

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

F

Free POU

A programmable object unit (POU), typically containing library functions, that can be programmed and updated independently of the master task of a program. Free POUs are available to be called from within programs as subroutines or jumps. For example, the *periodic task* is a subroutine that is implemented as a Free POU.

G

GRAFCET

The functioning of a sequential operation in a structured and graphic form.

This is an analytical method that divides any sequential control system into a series of steps, with which actions, transitions, and conditions are associated.

I

instruction list language

A program written in the instruction list language that is composed of a series of text-based instructions executed sequentially by the controller. Each instruction includes a line number, an instruction code, and an operand (see IEC 61131-3).

L

ladder diagram language

A graphical representation of the instructions of a controller program with symbols for contacts, coils, and blocks in a series of rungs executed sequentially by a controller (see IEC 61131-3).

M

master task

A processor task that is run through its programming software. The master task has 2 sections:

- **IN:** Inputs are copied to the IN section before execution of the master task.
- **OUT:** Outputs are copied to the OUT section after execution of the master task.

P

post configuration

(post configuration) An option that allows to modify some parameters of the application without changing the application. Post configuration parameters are defined in a file that is stored in the controller. They are overloading the configuration parameters of the application.

POU

(program organization unit) A variable declaration in source code and a corresponding instruction set. POU's facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POU's are available to one another.

S

symbol

A string of a maximum of 32 alphanumeric characters, of which the first character is alphabetic. It allows you to personalize a controller object to facilitate the maintainability of the application.

symbolic addressing

The indirect method of addressing memory objects, including physical inputs and outputs, used in programming instructions as operands and parameters by first defining symbols for them using these symbols in association with the programming instructions.

In contrast to immediate addressing, this is the preferred method because if the program configuration changes, symbols are automatically updated with their new immediate address associations. By contrast, any immediate addresses used as operands or parameters are not updated (refer to *immediate addressing*).

W

watchdog

A watchdog is a special timer used to ensure that programs do not overrun their allocated scan time. The watchdog timer is usually set to a higher value than the scan time and reset to 0 at the end of each scan cycle. If the watchdog timer reaches the preset value, for example, because the program is caught in an endless loop, an error is declared and the program stopped.



A

- accumulator, 153
- action zone, 137
- addressing
 - symbolic, 68
- allocating memory in controller, 70
- allocation mode, 70
- animation tables, 113
- application
 - behavior, configuring, 80
 - definition of, 24
 - downloading to controller, 203
 - protecting with a password, 55
 - protecting with password, 58
 - whether password-protected, 182
- assignment instructions
 - inserting in Ladder Diagram rungs, 145

B

- bill of material (BOM), printing, 51
- Boolean
 - accumulator, 153
- Boolean operators
 - graphic elements for, 140
- branching modes
 - graphic element, 139

C

- cache memory, consumption of, 129
- catalog, 61
 - replacing logic controller with reference from, 62
- coils
 - graphic elements for, 141
 - graphical representation of outputs, 135
- comments
 - adding to Instruction List, 152
 - adding to Ladder Diagrams, 146

- commissioning, 26
- comparison block
 - graphic elements for, 140
- comparison blocks
 - inserting IL expressions in, 144
- comparison expression
 - inserting in Ladder Diagram rungs, 144
- compile, date and time of last, 129
- configuration
 - current, 61
 - replacing logic controller in, 62
- configuring
 - application behavior, 80
 - master task, 96
 - periodic task duration, 102
 - project properties, 55
- contacts
 - graphic elements for, 139
 - graphical representation of inputs, 135
- controller time, displaying in trace, 170
- copying and pasting
 - POU, 94
- creating
 - Free POU, 93
- customizing, Ladder Editor, 49

D

- developing programs, stages of, 25
- development stages, 26
- digital inputs
 - configuring as event sources, 105
- downloading
 - application directly to controller, 39
 - firmware updates, 205
 - user application to controller, 203

E

- end/jump
 - graphic elements, 142

- event source
 - assigning subroutine as, 108
 - types of, 105
- event tasks
 - managing, 107
 - overview, 104
- events
 - since last cold restart, 109
 - triggering subroutines with, 105
- EXCEPTION state
 - fallback mode behavior, 82
- expansion modules
 - supported devices, 20
- exporting
 - symbol list, 126
 - trace, 171

F

- fallback
 - mode, specifying, 82
- firmware, downloading updates to controller, 205
- forcing values
 - in animation tables, 113
 - of I/Os, 182
- Free POU
 - assigning to an event source, 108
 - assigning to events, 94
 - assigning to periodic task, 94
- Free POU
 - creating, 93
- Free POU
 - for periodic task, 100
 - introduction to, 87
- function blocks
 - graphic element, 141
- functional levels, 80

G

- general settings, 49

- Grafcet, 162
 - how to use the instructions, 167
 - instructions, 162
 - post-processing, 166
 - preprocessing, 163
 - program structure, 163
 - sequential processing, 165
- grafcet instructions
 - graphic element, 142
- graphic elements
 - Ladder diagrams, 138
- grid lines, style of in Ladder Editor, 49

H

- hardware tree, 61

I

- importing
 - symbol list, 125
- inputs
 - configuring as event sources, 105
 - modifying, 147
- Instruction List
 - comments, 152
 - displaying symbols in, 69
 - using rung templates with, 131
- instructions
 - upstream/downstream, 147

K

- keyboard shortcuts, 49

L

- Ladder diagrams
 - comments, 146
 - graphic elements, 138
 - introduction, 135
 - programming principles, 137
 - reversing to Instruction List, 71
 - rungs, 136
 - using parentheses in, 148
 - using rung templates with, 131
- Ladder Editor
 - customizing, 49
 - defining symbols in, 69
 - resetting pointer after insertion, 49
- language, of user interface, 49
- life cycle state
 - of logic controller, 47
- line
 - graphic element, 139
- List instructions, 154
- List language
 - overview, 151
- logic controller
 - date and time last stopped, 182
 - displaying state, 182
 - downloading an application directly to, 39
 - replacing current in configuration, 62
 - state on startup, configuring, 81
 - supported types, 20
 - updating RTC of, 184

M

- master task
 - assigning POU as, 87
 - configuring, 96
 - system bits and words controlling, 97
- memory allocation, 70
- memory consumption, viewing, 129
- modem, displaying status of, 182
- modes, offline/online/simulator, 27
- module areas, 26

N

- normal scan mode, 97

O

- objects
 - definition of, 67
 - to trace in animation table, 113
 - updating values of in real time, 113
- offline mode
 - displayed in status area, 47
 - overview, 27
- online mode, 70
 - animation tables in, 113
 - displayed in status area, 47
 - editing values in animation table, 114
 - overview, 27
 - updating RTC in, 184
- operands, 153
- operation blocks
 - graphic element, 142
 - inserting assignment instructions in, 145
- operations
 - inserting in Ladder Diagram rungs, 145
- outputs
 - modifying, 147

P

- parentheses
 - modifiers, 159
 - nesting, 159
 - using in Ladder diagrams, 148
 - using in programs, 158
- password
 - protecting an application , 58
 - removing from application, 58
 - removing from project, 57
 - requiring to open project file, 57
 - whether application is protected with, 182
- password-protecting an application, 55
- period, scan, 97

- periodic
 - scan mode, 97
 - scan period, 102
 - tasks, 100
- periodic task
 - as event source, 105
 - assigning Free POU to, 94
 - configuring duration of, 102
- POU
 - copying, 94
 - Free, 100
 - pasting, 94
- printing reports, 51
- priority level, of events, 104
- program
 - compiling, 45
 - definition of, 24
 - displaying number of lines in, 129
 - jumps, 147
- program development, stages of, 25
- program organization unit (POU), 87
- program, configuring fallback modes, 82
- programming
 - best practices, 147
 - grid, 137
 - languages, supported, 22
- project
 - configuring properties, 55
 - definition of, 24
 - displaying report for, 51
 - protecting with password, 57
 - templates, 38
- properties, 55
- pulse width (TON) , 97

R

- RAM
 - executable contains application, 182
- RAM memory, consumption of, 129
- relay circuits, representing as Ladder diagrams, 135
- removing password protection, 57, 58
- replacing
 - logic controller in configuration, 62

- reports
 - exporting, 51
 - printing, 51
- reversibility
 - introduction to, 71
- RTC
 - displaying date and time, 182
 - managing with system bits, 147
 - updating in controller, 184
- rung templates, 131
- rungs
 - graphic element, 138

S

- scan modes, 83, 97
- scan task, configuring watchdog, 82
- scan time
 - displaying minimum, maximum, current, 182
 - minimum, displayed in status area, 47
- sections
 - in events, 104
 - of master task, 96
- selection
 - graphic element, 139
- settings
 - general, 49
- sharing
 - symbol list, 127
- sharing symbols
 - with Vijeo Designer project, 127
- simulator, 186
 - accessing the simulator, 186
 - how to use, 198
 - I/O manager window, 188
 - modifying values, 193
 - modifying values of analog inputs, 194
 - modifying values of digital inputs, 193
 - output tracing, 195
 - simulator windows, 186
 - Time Management window, 190
- simulator mode
 - overview, 27
- sources of events, 105

stages of developing a program, 26
 Start Page, 26
 startup state of logic controller, 81
 state

- initial logic controller, configuring, 81
- of controller, displaying, 182

 status area, 47
 stop sensors, wiring, 147
 STOP state

- fallback mode behavior, 82

 subroutine

- assigning to periodic task, 100
- assigning to tasks, 107
- associated with periodic event, 105
- implementing as Free POU, 87
- of master task, 96
- triggering execution with an event, 105

 supported devices, 20
 symbol list

- displaying, 125
- exporting, 126
- importing, 125
- sharing with Vijeo Designer project, 127

 symbolic addressing, 68
 symbols

- addressing with, 68
- defining in graphic elements of Ladder editor, 69
- defining in Properties window, 68
- displaying in Instruction List code, 69
- list of used, 125
- storing in logic controller, 69

 system bits

- %S0, 147
- %S11, 97
- %S19, 97
- %S31, 109
- %S38, 109
- %S39, 109
- %S9, 147

 system bits/words

- controlling events with, 109
- in symbol list, 125

system words

- %SW0, 97
- %SW27, 97
- %SW30, 97
- %SW30...%SW32, 182
- %SW31, 97
- %SW32, 97
- %SW48, 109
- %SW54...%SW57, 182

T

task

- event, 104
- periodic, 100

 tasks, 83
 tasks and scan modes, 83
 template

- inserting into rung, 131
- project, 38

 test zone, 137
 TH0, TH1

- configuring as event sources, 105

 threshold outputs (of %HSC)

- configuring as event sources, 105

 time base (for trace), 115
 timer, watchdog, 82
 toolbar buttons, 45
 trace

- displaying, 170
- exporting to PDF, 171
- selecting objects to, 113
- selecting time base for, 115

 trace window

- displaying, 115

U

uploading

- preventing with a password, 58

 user interface, setting language of, 49

W

watchdog timer, configuring, 82

wiring stop sensors, *147*

X

XOR

graphic elements for, *140*