

# Modicon M221

## Logic Controller

### Programming Guide

12/2015



SoMachine Basic

---

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

---



	<b>Safety Information</b> .....	<b>7</b>
	<b>About the Book</b> .....	<b>9</b>
<b>Part I</b>	<b>Introduction</b> .....	<b>17</b>
<b>Chapter 1</b>	<b>About the Modicon M221 Logic Controller</b> .....	<b>19</b>
	TM221C Logic Controller Description .....	<b>20</b>
	TM221M Logic Controller Description .....	<b>25</b>
<b>Chapter 2</b>	<b>Configuration Features</b> .....	<b>29</b>
2.1	Objects .....	<b>30</b>
	Objects .....	<b>31</b>
	Object Types .....	<b>32</b>
	Addressing I/O Objects .....	<b>36</b>
	Maximum Number of Objects .....	<b>38</b>
2.2	Task Structure .....	<b>42</b>
	Tasks and Scan Modes .....	<b>43</b>
	Maximum Number of Tasks and Priorities .....	<b>46</b>
2.3	Controller States and Behaviors .....	<b>47</b>
	Controller States Diagram .....	<b>48</b>
	Controller States Description .....	<b>49</b>
	Controller State Transitions .....	<b>53</b>
	Persistent Variables .....	<b>56</b>
	Output Behavior .....	<b>57</b>
2.4	Post Configuration .....	<b>61</b>
	Post Configuration .....	<b>62</b>
	Post Configuration File Management .....	<b>63</b>
<b>Part II</b>	<b>Configuring the M221 Logic Controller</b> .....	<b>65</b>
<b>Chapter 3</b>	<b>How to Configure a Controller</b> .....	<b>67</b>
	Building a Configuration .....	<b>68</b>
	Optional I/O Expansion Modules .....	<b>73</b>
	Configuring the M221 Logic Controller .....	<b>77</b>
	Updating Firmware using Executive Loader Wizard .....	<b>78</b>
<b>Chapter 4</b>	<b>Embedded Input/Output Configuration</b> .....	<b>79</b>
4.1	Digital Input Configuration .....	<b>80</b>
	Configuring Digital Inputs .....	<b>80</b>

4.2	Digital Output Configuration . . . . .	84
	Configuring Digital Outputs . . . . .	84
4.3	Analog Input Configuration. . . . .	86
	Configuring Analog Inputs . . . . .	86
4.4	High Speed Counter Configuration . . . . .	88
	Configuring High Speed Counters . . . . .	89
	Configuring Dual Phase and Single Phase Counters . . . . .	92
	Configuring Frequency Meter. . . . .	96
4.5	Pulse Generator Configuration. . . . .	98
	Configuring Pulse Generators . . . . .	99
	Configuring Pulse (%PLS). . . . .	102
	Configuring Pulse Width Modulation (%PWM). . . . .	104
	Configuring Pulse Train Output (%PTO) . . . . .	106
	Configuring Frequency Generator (%FREQGEN). . . . .	109
<b>Chapter 5</b>	<b>I/O Bus Configuration . . . . .</b>	<b>111</b>
	I/O Configuration General Practices . . . . .	112
	Maximum Hardware Configuration. . . . .	113
	Configuring Cartridges and Expansion Modules . . . . .	117
<b>Chapter 6</b>	<b>Embedded Communication Configuration . . . . .</b>	<b>119</b>
6.1	Ethernet Configuration . . . . .	120
	Configuring Ethernet Network . . . . .	121
	Configuring Modbus TCP. . . . .	126
	Configuring EtherNet/IP . . . . .	130
6.2	Serial Line Configuration . . . . .	133
	Configuring Serial Line. . . . .	133
6.3	Supported Modbus Function Codes. . . . .	141
	Supported Modbus Function Codes. . . . .	141
<b>Chapter 7</b>	<b>SD Card . . . . .</b>	<b>143</b>
	File Management Operations. . . . .	144
	SD Card Supported File Types . . . . .	146
	Clone Management . . . . .	148
	Firmware Management . . . . .	150
	Application Management . . . . .	152
	Post Configuration Management . . . . .	154
	Error Log Management . . . . .	156
	Memory Management: Backing Up and Restoring Controller Memory . . . . .	159

---

<b>Part III</b>	<b>Programming the M221 Logic Controller . . . . .</b>	<b>161</b>
<b>Chapter 8</b>	<b>I/O Objects. . . . .</b>	<b>163</b>
	Digital Inputs (%I) . . . . .	164
	Digital Outputs (%Q) . . . . .	165
	Analog Inputs (%IW) . . . . .	166
	Analog Outputs (%QW) . . . . .	168
<b>Chapter 9</b>	<b>Network Objects . . . . .</b>	<b>169</b>
	Input Assembly Objects (%QWE) . . . . .	170
	Output Assembly Objects (%IWE) . . . . .	172
	Input Registers Objects (%QWM) . . . . .	173
	Output Registers Objects (%IWM) . . . . .	175
<b>Chapter 10</b>	<b>Function Blocks . . . . .</b>	<b>177</b>
	Advanced Expert Input/Output Functions . . . . .	177
<b>Chapter 11</b>	<b>System Objects. . . . .</b>	<b>179</b>
	System Bits (%S) . . . . .	180
	System Words (%SW) . . . . .	188
<b>Glossary</b>	. . . . .	<b>211</b>
<b>Index</b>	. . . . .	<b>217</b>



---

# 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, service, 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 document describes the configuration and programming of the Modicon M221 Logic Controller for SoMachine Basic. For further information, refer to the separate documents provided in the SoMachine Basic online help.

### Validity Note

This document has been updated for the release of SoMachine Basic V1.4.

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 <a href="http://www.schneider-electric.com">www.schneider-electric.com</a> .
2	In the <b>Search</b> box type the reference of a product or the name of a product range. <ul style="list-style-type: none"><li>● Do not include blank spaces in the reference or product range.</li><li>● To get information on grouping similar modules, use asterisks (*).</li></ul>
3	If you entered a reference, go to the <b>Product Datasheets</b> search results and click on the reference that interests you. If you entered the name of a product range, go to the <b>Product Ranges</b> search results and click on the product range that interests you.
4	If more than one reference appears in the <b>Products</b> 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 <b>Download XXX product datasheet</b> .

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 - Operating Guide	EIO0000001354 (ENG) EIO0000001355 (FRA) EIO0000001356 (GER) EIO0000001357 (SPA) EIO0000001358 (ITA) EIO0000001359 (CHS) EIO0000001366 (POR) EIO0000001367 (TUR)
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 (FRA) EIO0000002009(GER) EIO0000002010 (SPA) EIO0000002011 (ITA) EIO0000002012 (CHS) EIO0000002013 (POR) EIO0000002014 (TUR)
Modicon M221 Logic Controller - Hardware Guide	EIO0000001384 (ENG) EIO0000001385 (FRA) EIO0000001386 (GER) EIO0000001387 (SPA) EIO0000001388 (ITA) EIO0000001389 (CHS) EIO0000001370 (POR) EIO0000001371 (TUR)
TMH2GDB Remote Graphic Display - User Guide	EIO0000002063 (ENG) EIO0000002064 (FRA) EIO0000002065 (GER) EIO0000002066 (SPA) EIO0000002067 (ITA) EIO0000002068 (CHS) EIO0000002069 (POR) EIO0000002070 (TUR)

Title of Documentation	Reference Number
Modicon TMC2 Cartridge - Programming Guide	EIO0000001782 (ENG) EIO0000001783 (FRA) EIO0000001784 (GER) EIO0000001785 (SPA) EIO0000001786 (ITA) EIO0000001787 (CHS) EIO0000001788 (POR) EIO0000001789 (TUR)
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)

Title of Documentation	Reference Number
Modicon TM3 Expert Modules - Hardware Guide	EIO0000001420 (ENG) EIO0000001421 (FRA) EIO0000001422 (GER) EIO0000001423 (SPA) EIO0000001424 (ITA) EIO0000001425 (CHS) EIO0000001380 (POR) EIO0000001381 (TUR)
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)

Title of Documentation	Reference Number
Modicon TM2 Analog I/O Modules - Hardware Guide	EIO0000000034 (ENG) EIO0000000035 (FRA) EIO0000000036 (GER) EIO0000000037 (SPA) EIO0000000038 (ITA) EIO0000000039 (CHS)
SR2MOD02 and SR2MOD03 Wireless Modem - User Guide	EIO0000001575 (ENG)

You can download these technical publications and other technical information from our website at <http://download.schneider-electric.com>

### Product Related Information

 <b>WARNING</b>
<p><b>LOSS OF CONTROL</b></p> <ul style="list-style-type: none"> <li>● 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.</li> <li>● Separate or redundant control paths must be provided for critical control functions.</li> <li>● System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.</li> <li>● Observe all accident prevention regulations and local safety guidelines.<sup>1</sup></li> <li>● Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.</li> </ul> <p><b>Failure to follow these instructions can result in death, serious injury, or equipment damage.</b></p>

<sup>1</sup> 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.**

### Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
EN 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2008	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
EN 1088:2008 ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2006	Safety of machinery - Emergency stop - Principles for design
EN/IEC 62061:2005	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.

Standard	Description
IEC 61784-3:2008	Digital data communication for measurement and control: Functional safety field buses.
2006/42/EC	Machinery Directive
2004/108/EC	Electromagnetic Compatibility Directive
2006/95/EC	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *EC Machinery Directive (EC/2006/42)* and *ISO 12100:2010*.

**NOTE:** The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.



---

# Part I

## Introduction

---

### Overview

This part provides general information about the Modicon M221 Logic Controller and its configuration and programming features.

### What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	About the Modicon M221 Logic Controller	19
2	Configuration Features	29



---

# Chapter 1

## About the Modicon M221 Logic Controller

---

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
TM221C Logic Controller Description	20
TM221M Logic Controller Description	25

## TM221C Logic Controller Description

### Overview

The TM221C Logic Controller has various powerful features and can service a wide range of applications.

Software configuration, programming, and commissioning are accomplished with the SoMachine Basic software described in the SoMachine Basic Operating Guide and the M221 Logic Controller - Programming Guide.

### Programming Languages

The M221 Logic Controller is configured and programmed with the SoMachine Basic software, which supports the following IEC 61131-3 programming languages:

- IL: Instruction List
- LD: Ladder Diagram
- Grafset (List)
- Grafset (SFC)

### Power Supply

The power supply of the TM221C Logic Controller is 24 Vdc or 100...240 Vac.

### Real Time Clock

The M221 Logic Controller includes a Real Time Clock (RTC) system.

### Run/Stop

The M221 Logic Controller can be operated externally by the following:

- a hardware Run/Stop switch
- a Run/Stop operation by a dedicated digital input, defined in the software configuration (for more information, refer to Configuring Digital Inputs ([see page 80](#)).)
- SoMachine Basic software (for more information, refer to Toolbar ([see SoMachine Basic, Operating Guide](#))).
- a TMH2GDB Remote Graphic Display (for more information, refer to Controller State Menu ([see Modicon TMH2GDB, Remote Graphic Display, User Guide](#))).

## Memory

This table describes the different types of memory:

Memory Type	Size	Used to
RAM	512 Kbytes of RAM memory: 256 Kbytes for internal variables and 256 Kbytes for application and data.	execute the application and contain data
Non-volatile	1.5 Mbytes, of which 256 Kbytes is used to back up the application and data in case of power outage.	save the application

## Embedded Inputs/Outputs

The following embedded I/O types are available, depending on the controller reference:

- Regular inputs
- Fast inputs associated with counters
- Regular sink/source transistor outputs
- Fast sink/source transistor outputs associated with pulse generators
- Relay outputs
- Analog inputs

## Removable Storage

The M221 Logic Controllers include an embedded SD card slot.

The Modicon M221 Logic Controller allows the following types of file management with an SD card:

- Clone management (*see page 148*): back up the application, firmware, and post configuration (if it exists) of the logic controller
- Firmware management (*see page 150*): download firmware updates directly to the logic controller, and download firmware to a TMH2GDB Remote Graphic Display
- Application management (*see page 152*): back up and restore the logic controller application, or copy it to another logic controller of the same reference
- Post configuration management (*see page 154*): add, change, or delete the post configuration file of the logic controller
- Error log management (*see page 156*): back up or delete the error log file of the logic controller
- Memory management: backup/restore of memory bits and words from a controller

## Embedded Communication Features

The following types of communication ports are available depending on the controller reference:

- Ethernet
- USB Mini-B
- Serial Line 1

## Remote Graphic Display

For more information, refer to the Modicon TMH2GDB Remote Graphic Display - User Guide.

## TM221C Logic Controller

Reference	Digital Inputs	Digital Outputs	Analog Inputs	Communication Ports	Power Supply
TM221C16R	5 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	7 relay outputs	Yes	1 serial line port 1 USB programming port	100...240 Vac
TM221CE16R			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C16T	5 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	Source outputs 5 regular transistor outputs 2 fast outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE16T			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C16U	5 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	Sink outputs 5 regular transistor outputs 2 fast outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE16U				1 serial line port 1 USB programming port 1 Ethernet port	
TM221C24R	10 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	10 relay outputs	Yes	1 serial line port 1 USB programming port	100...240 Vac
TM221CE24R			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C24T	10 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	Source outputs 8 regular transistor outputs 2 fast outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE24T			Yes	1 serial line port 1 USB programming port 1 Ethernet port	

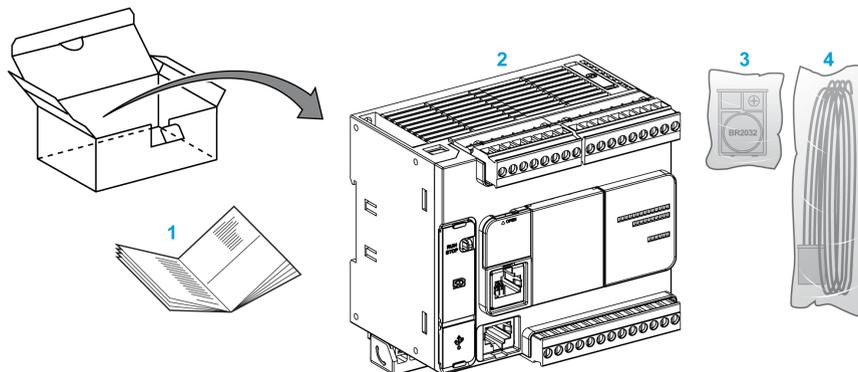
**NOTE:** The TM221C Logic Controller uses removable screw terminal blocks.

(1) The regular inputs have a maximum frequency of 5 kHz.  
(2) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.  
(3) The fast transistor outputs can be used either as regular transistor outputs, or for PLS, PWM or PTO functions, or reflex outputs for HSC.

Reference	Digital Inputs	Digital Outputs	Analog Inputs	Communication Ports	Power Supply
TM221C24U	10 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	Sink outputs 8 regular transistor outputs 2 fast outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE24U (see <i>Modicon M221 Logic Controller, Hardware Guide</i> )			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C40R	20 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	16 relay outputs	Yes	1 serial line port 1 USB programming port	100...240 Vac
TM221CE40R			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C40T		Source outputs 14 regular transistor outputs 2 fast outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE40T			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C40U	20 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	Sink outputs 12 regular transistor outputs 4 fast outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE40U (see <i>Modicon M221 Logic Controller, Hardware Guide</i> )			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
<b>NOTE:</b> The TM221C Logic Controller uses removable screw terminal blocks.					
(1) The regular inputs have a maximum frequency of 5 kHz.					
(2) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.					
(3) The fast transistor outputs can be used either as regular transistor outputs, or for PLS, PWM or PTO functions, or reflex outputs for HSC.					

## Delivery Content

The following figure presents the content of the delivery for a TM221C Logic Controller:



- 1 TM221C Logic Controller Instruction Sheet
- 2 TM221C Logic Controller
- 3 Battery holder with lithium carbon monofluoride battery, type Panasonic BR2032.
- 4 Analog cable

## TM221M Logic Controller Description

### Overview

The TM221M Logic Controller has various powerful features and can service a wide range of applications.

Software configuration, programming, and commissioning are accomplished with the SoMachine Basic software described in the SoMachine Basic Operating Guide and the M221 Logic Controller - Programming Guide.

### Programming Languages

The M221 Logic Controller is configured and programmed with the SoMachine Basic software, which supports the following IEC 61131-3 programming languages:

- IL: Instruction List
- LD: Ladder Diagram
- Grafcet (List)
- Grafcet (SFC)

### Power Supply

The power supply of the TM221M Logic Controller is 24 Vdc.

### Real Time Clock

The M221 Logic Controller includes a Real Time Clock (RTC) system.

### Run/Stop

The M221 Logic Controller can be operated externally by the following:

- a hardware Run/Stop switch
- a Run/Stop operation by a dedicated digital input, defined in the software configuration (for more information, refer to Configuring Digital Inputs ([see page 80](#)))
- SoMachine Basic software (for more information, refer to Toolbar ([see SoMachine Basic, Operating Guide](#))).
- a TMH2GDB Remote Graphic Display (for more information, refer to Controller State Menu).

## Memory

This table describes the different types of memory:

Memory Type	Size	Used to
RAM	512 Kbytes of RAM memory: 256 Kbytes for internal variables and 256 Kbytes for application and data.	execute the application and contains data
Non-volatile	1.5 Mbytes, of which 256 Kbytes is used to back up the application and data in case of power outage.	save the application

## Embedded Inputs/Outputs

The following embedded I/O types are available, depending on the controller reference:

- Regular inputs
- Fast inputs (HSC)
- Regular transistor outputs
- Fast transistor outputs (PLS/PWM/PTO)
- Relay outputs
- Analog inputs

## Removable Storage

The M221 Logic Controllers include an embedded SD card slot.

The Modicon M221 Logic Controller allows the following types of file management with an SD card:

- Clone management ([see page 148](#)): back up the application, firmware, and post configuration (if it exists) of the logic controller
- Firmware management ([see page 150](#)): download firmware updates directly to the logic controller, and download firmware to a TMH2GDB Remote Graphic Display
- Application management ([see page 152](#)): back up and restore the logic controller application, or copy it to another logic controller of the same reference
- Post configuration management ([see page 154](#)): add, change, or delete the post configuration file of the logic controller
- Error log management ([see page 156](#)): back up or delete the error log file of the logic controller
- Memory management: backup/restore of memory bits and words from a controller

## Embedded Communication Features

The following communication ports are available on the front panel of the controller, depending on the controller reference:

- Ethernet
- USB Mini-B
- SD Card
- Serial Line 1
- Serial Line 2

## Remote Graphic Display

For more information, refer to the Modicon TMH2GDB Remote Graphic Display - User Guide.

## TM221M Logic Controller

Reference	Digital Input	Digital Output	Analog Input	Communication Ports	Terminal Type
TM221M16R	4 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	8 relay outputs	Yes	2 serial line ports 1 USB programming port	Removable screw terminal blocks
TM221M16RG	4 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	8 relay outputs	Yes	2 serial line ports 1 USB programming port	Removable spring terminal blocks
TM221ME16R	4 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	8 relay outputs	Yes	1 serial line port 1 USB programming port 1 Ethernet port	Removable screw terminal blocks
TM221ME16RG	4 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	8 relay outputs	Yes	1 serial line port 1 USB programming port 1 Ethernet port	Removable spring terminal blocks
TM221M16T	4 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	6 regular transistor outputs 2 fast transistor outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	2 serial line ports 1 USB programming port	Removable screw terminal blocks
TM221M16TG	4 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	6 regular transistor outputs 2 fast transistor outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	2 serial line ports 1 USB programming port	Removable spring terminal blocks
TM221ME16T	4 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	6 regular transistor outputs 2 fast transistor outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	1 serial line port 1 USB programming port 1 Ethernet port	Removable screw terminal blocks

**NOTE:** The TM221M Logic Controller uses a 24 Vdc power supply.

**(1)** The regular inputs I2, I3, I4, and I5 have a maximum frequency of 5 kHz.

The other regular inputs have a maximum frequency of 100 Hz.

**(2)** The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.

**(3)** The fast transistor outputs can be used as regular transistor outputs, or for PLS, PWM or PTO functions, or reflex outputs for HSC.

Reference	Digital Input	Digital Output	Analog Input	Communication Ports	Terminal Type
TM221ME16TG	4 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	6 regular transistor outputs 2 fast transistor outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	1 serial line port USB programming port 1 Ethernet port	Removable spring terminal blocks
TM221M32TK	12 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	14 regular transistor outputs 2 fast outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	2 serial line ports 1 USB programming port	HE10 (MIL 20) connectors
TM221ME32TK	12 regular inputs <sup>(1)</sup> 4 fast inputs (HSC) <sup>(2)</sup>	14 regular outputs 2 fast outputs (PLS/PWM/PTO) <sup>(3)</sup>	Yes	1 serial line port 1 USB programming port 1 Ethernet port	HE10 (MIL 20) connectors

**NOTE:** The TM221M Logic Controller uses a 24 Vdc power supply.

(1) The regular inputs I2, I3, I4, and I5 have a maximum frequency of 5 kHz.

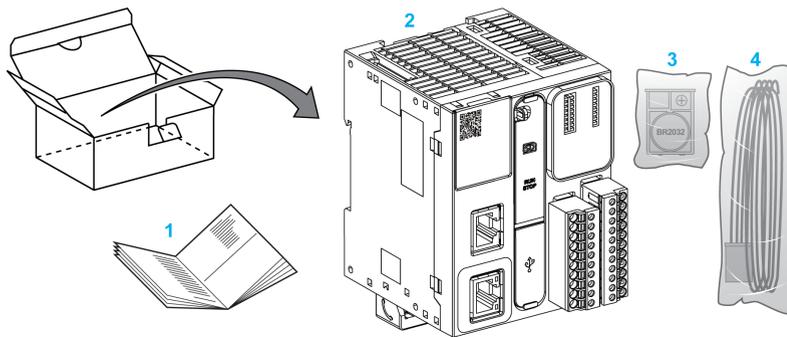
The other regular inputs have a maximum frequency of 100 Hz.

(2) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.

(3) The fast transistor outputs can be used as regular transistor outputs, or for PLS, PWM or PTO functions, or reflex outputs for HSC.

### Delivery Content

The following figure presents the content of the delivery for a TM221M Logic Controller:



- 1 TM221M Logic Controller Instruction Sheet
- 2 TM221M Logic Controller
- 3 Battery holder with lithium carbon monofluoride battery, type Panasonic BR2032.
- 4 Analog cable

---

# Chapter 2

## Configuration Features

---

### Introduction

This chapter provides information related to M221 Logic Controller memory mapping, task, states, behaviors, objects, and functions. The topics explained in this chapter allow the operator to understand the featured specifications of M221 Logic Controller that are primarily needed to configure and program the controller in SoMachine Basic.

### What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	Objects	30
2.2	Task Structure	42
2.3	Controller States and Behaviors	47
2.4	Post Configuration	61

# Section 2.1

## Objects

---

### What Is in This Section?

This section contains the following topics:

Topic	Page
Objects	31
Object Types	32
Addressing I/O Objects	36
Maximum Number of Objects	38

## 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.

## Object Types

### Introduction

The language object types for the M221 Logic Controller are described in the following table:

Object Type	Object	Object Function	Description
Memory objects	%M	Memory bits	Stores memory bit.
	%MW	Memory words	Stores 16-bit memory word.
	%MD	Memory double words	Stores 32-bit memory word.
	%MF	Memory floating point	Stores memory floating point in a mathematical argument which has a decimal in its expression.
	%KW	Constant words	Stores 16-bit constant word.
	%KD	Constant double words	Stores 32-bit constant word.
	%KF	Constant floating points	Stores constant floating point in a mathematical argument which has a decimal in its expression.
System objects	%S	System bits <i>(see page 180)</i>	Stores system bit.
	%SW	System words <i>(see page 188)</i>	Stores system word.

Object Type	Object	Object Function	Description
I/O objects	%I	Input bits (see page 164)	Stores value of the digital input.
	%Q	Output bits (see page 165)	Stores value of the digital output.
	%IW	Input words (see page 166)	Stores value of the analog input.
	%QW	Output words (see page 168)	Stores value of the analog output.
	%FC	Fast counters	Execute fast counts of pulses from sensors, switches, and so on.
	%HSC	High speed counters	Execute fast counts of pulses from sensors, switches, and so on, that are connected to the fast inputs.
	%PLS	Pulse	Generates a square wave pulse signal on dedicated output channels.
	%PWM	Pulse width modulation	Generates a modulated wave signal on dedicated output channels with a variable duty cycle.
	%PTO	Pulse train output (see <i>Modicon M221 Logic Controller, Advanced Functions Library Guide</i> )	Generates a pulse train output to control a linear single-axis stepper or servo drive in open loop mode.
	%FREQGEN	Frequency Generator	Generates a square wave signal on a dedicated output channel with programmable frequency and duty cycle of 50%.
Network objects	%QWE	Input assembly (see page 170)	The values of EtherNet/IP Input assembly frames sent by the logic controller. <b>NOTE:</b> For more details about directionality, refer to Configuring EtherNet/IP (see page 131).
	%IWE	Output assembly (see page 172)	The values of EtherNet/IP Output assembly frames received by the logic controller. <b>NOTE:</b> For more details about directionality, refer to Configuring EtherNet/IP (see page 131).
	%QWM	Input registers (see page 173)	The values of Modbus mapping table Input registers sent by the logic controller.
	%IWM	Output registers (see page 175)	The values of Modbus mapping table Output registers received by the logic controller.

Object Type	Object	Object Function	Description
Software objects	%TM	Timers	Specifies a time before triggering an action.
	%C	Counters	Provides up and down counting of actions.
	%MSG	Messages	Stores the status message at the communication port.
	%R	LIFO/FIFO registers	Stores memory up to 16 words of 16 bits each in 2 different ways, queue, and stacks.
	%DR	Drums	Operates on a principle similar to an electromechanical drum controller which changes step according to external events.
	%SBR	Shift bit registers	Provides a left or right shift of binary data bits (0 or 1).
	%SC	Step counters	Provides a series of steps to which actions can be assigned.
	SCH	Schedule blocks	Controls actions at a predefined month, day, and time.
	PID	PID	Provides a generic control loop feedback in which output is proportional, integral, and derivative of the input.
PTO objects	Refer to Pulse Train Output ( <i>see Modicon M221 Logic Controller, Advanced Functions Library Guide</i> ).		
Communication objects	%READ_VAR	Read Var	The %READ_VAR function block is used to read data from a remote device on Modbus SL or Modbus TCP.
	%WRITE_VAR	Write Var	The %WRITE_VAR function block is used to write data to an external device using the Modbus SL or Modbus TCP protocol.
	%WRITE_READ_VAR	Write Read Var	The %WRITE_READ_VAR function block is used to read and write data stored in internal memory words to an external device using the Modbus SL or Modbus TCP protocol.
	%SEND_RECV_MSG	Send Receive Message	The %SEND_RECV_MSG function block is used to send or receive data on a serial line configured for the ASCII protocol.
	%SEND_RECV_SMS	Send Receive SMS ( <i>see SoMachine Basic, Generic Functions Library Guide</i> )	The %SEND_RECV_SMS function block is used to send or receive SMS messages through a GSM modem connected to a serial line.

Memory objects and software objects are generic objects used in SoMachine Basic, whereas system objects and I/O objects are controller-specific. All controller-specific objects are discussed in the Programming ([see page 161](#)) section.

For programming details of memory objects, software objects, and communication objects, refer to the SoMachine Basic Generic Functions Library Guide.

For programming details of PID objects and PTO objects, refer to the Advanced Functions Library Guide.

## Addressing I/O Objects

### Addressing Examples

This table presents addressing examples for various object types:

Object Type	Syntax	Example	Description
<b>Memory objects</b>			
Memory bits	%M <i>i</i>	%M25	Internal memory bit 25.
Memory words	%MW <i>i</i>	%MW15	Internal memory word 15.
Memory double words	%MD <i>i</i>	%MD16	Internal memory double word 16.
Memory floating points	%MF <i>i</i>	%MF17	Internal memory floating point 17.
Constant words	%KW <i>i</i>	%KW26	Constant word 26.
Constant double words	%KD <i>i</i>	%KD27	Internal constant double word 27.
Constant floating points	%KF <i>i</i>	%KF28	Internal constant floating point 28.
<b>System objects</b>			
System bits	%S <i>i</i>	%S8	System bit 8.
System words	%SW <i>i</i>	%SW30	System word 30.
<b>I/O objects</b>			
Digital inputs	%I <i>y.z</i>	%I0.5	Digital input 5 on the controller (embedded I/O).
Digital outputs	%Q <i>y.z</i>	%Q3.4	Digital output 4 on the expansion module at address 3 (expansion module I/O).
Analog inputs	%IW <i>y.z</i>	%IW0.1	Analog input 1 on the controller (embedded I/O).
Analog outputs	%QW0. <i>mOn</i>	%QW0.100	Analog output 0 on the cartridge 1.
Fast counters	%FC <i>i</i>	%FC2	Fast counter 2 on the controller.
High speed counters	%HSC <i>i</i>	%HSC1	High speed counter 1 on the controller.
Pulse	%PLS <i>i</i>	%PLS0	Pulse output 0 on the controller.
Pulse width modulation	%PWM <i>i</i>	%PWM1	Pulse width modulation output 1 on the controller.
Pulse train output	%PTO <i>i</i>	%PTO1	Pulse train output 1 on the controller.
<b>Network objects</b>			
Input assembly	%QWE <i>i</i>	%QWE8	Input assembly instance 8.
<p><b>i</b> Object instance identifier that indicates the instance of the object on the controller.  <b>m</b> Cartridge number on the controller.  <b>n</b> Channel number on the cartridge.  <b>y</b> Indicates the I/O type. It is 0 for the controller and 1, 2, and so on, for the expansion modules.  <b>z</b> Channel number on the controller or expansion module.</p>			

Object Type	Syntax	Example	Description
Output assembly	%IWE <i>i</i>	%IWE6	Output assembly instance 6.
Input registers	%QWM <i>i</i>	%QWM1	Input register instance 1.
Output registers	%IWM <i>i</i>	%IWM0	Output register instance 0.
<b>Software objects</b>			
Timers	%TM <i>i</i>	%TM5	Timer instance 5.
Counters	%C <i>i</i>	%C2	Counter instance 2.
Message	%MSG <i>i</i>	%MSG1	Program compilation status message 1.
LIFO/FIFO registers	%R <i>i</i>	%R3	FIFO/LIFO registers instance 3.
Drums	%DR <i>i</i>	%DR6	Drum register 6 on the controller.
Shift bit registers	%SBR <i>i</i>	%SBR5	Shift bit register 5 on the controller.
Step counters	%SC <i>i</i>	%SC5	Step counter 5 on the controller.
Schedule blocks	SCH <i>i</i>	SCH 3	Schedule block 3 on the controller.
PID	PID <i>i</i>	PID 7	PID feedback object 7 on the controller.
<b>PTO objects</b>			
MC_Power_PTO (motion function block)	%MC_POWER_ PTO <i>i</i>	%MC_POWER_PTO1	MC_POWER_PTO function block instance 1.
MC_Reset_PTO (administrative function block)	%MC_RESET_ PTO <i>i</i>	%MC_RESET_PTO0	MC_RESET_PTO function block instance 0.
<b>Communication objects</b>			
Read Var	%READ_VAR <i>i</i>	%READ_VAR2	READ_VAR function block instance 2.
Write Var	%WRITE_VAR <i>i</i>	%WRITE_VAR4	WRITE_VAR function block instance 4.
Write Read Var	%WRITE_READ_ VAR <i>i</i>	%WRITE_READ_VAR0	WRITE_READ_VAR function block instance 0.
Send Receive Message	%SEND_RECV_M SG <i>i</i>	%SEND_RECV_MSG6	SEND_RECV_MSG function block instance 6.
Send Receive SMS	%SEND_RECV_S MS <i>i</i>	%SEND_RECV_SMS0	SEND_RECV_SMS function block instance 0.
<p><b>i</b> Object instance identifier that indicates the instance of the object on the controller.</p> <p><b>m</b> Cartridge number on the controller.</p> <p><b>n</b> Channel number on the cartridge.</p> <p><b>y</b> Indicates the I/O type. It is 0 for the controller and 1, 2, and so on, for the expansion modules.</p> <p><b>z</b> Channel number on the controller or expansion module.</p>			

## Maximum Number of Objects

### Maximum Number of Objects Description

This table provides information about the maximum number of objects supported by the M221 Logic Controller:

Objects	M221 Logic Controller References			
	Modular References		Compact References	
	TM221M16R• TM221ME16R•	TM221M16T• TM221ME16T• TM221M32TK TM221ME32TK	TM221C••R TM221CE••R	TM221C••T TM221CE••T TM221C••U TM221CE••U
<b>Memory objects</b>				
%M	512* 1024	512* 1024	512* 1024	512* 1024
%MW	8000	8000	8000	8000
%MD %MF	7999	7999	7999	7999
%KW	512	512	512	512
%KD %KF	511	511	511	511
<b>System objects</b>				
%S	160	160	160	160
%SW	234	234	234	234
<b>I/O objects</b>				
%I	8	8 (for TM221M16T• and TM221ME16T•)	9 (for TM221C16• and TM221CE16•)	9 (for TM221C16• and TM221CE16•)
		16 (for TM221M32TK and TM221ME32TK)	14 (for TM221C24• and TM221CE24•)	14 (for TM221C24• and TM221CE24•)
			24 (for TM221C40• and TM221CE40•)	24 (for TM221C40• and TM221CE40•)
* for firmware and software version < 1.3				

Objects	M221 Logic Controller References			
	Modular References		Compact References	
	TM221M16R• TM221ME16R•	TM221M16T• TM221ME16T• TM221M32TK TM221ME32TK	TM221C••R TM221CE••R	TM221C••T TM221CE••T TM221C••U TM221CE••U
%Q	8	8 (for TM221M16T• and TM221ME16T•)  16 (for TM221M32TK and TM221ME32TK)	7 (for TM221C16• and TM221CE16•)  10 (for TM221C24• and TM221CE24•)  16 (for TM221C40• and TM221CE40•)	7 (for TM221C16• and TM221CE16•)  10 (for TM221C24• and TM221CE24•)  16 (for TM221C40• and TM221CE40•)
%IW	2	2	2	2
%QW	0	0	<b>NOTE:</b> Analog outputs are not built in with the controller. Use cartridges TMC2AQ2V and/or TMC2AQ2C to add analog outputs to your controller configuration.  2 (if 1 cartridge is used) 4 (if 2 cartridges are used with TM221C40R or TM221CE40R)	
%FC	4	4	4	4
%HSC	Up to 4	Up to 4	Up to 4	Up to 4
%PLS %PWM %PTO %FREQGEN	0	2	0	2
<b>Network objects</b>				
%QWE	20 (for TM221ME16R•)	20 (for TM221ME16T• and TM221ME32TK)	20 (for TM221CE16•)	20 (for TM221CE16•)
* for firmware and software version < 1.3				

Objects	M221 Logic Controller References			
	Modular References		Compact References	
	TM221M16R• TM221ME16R•	TM221M16T• TM221ME16T• TM221M32TK TM221ME32TK	TM221C••R TM221CE••R	TM221C••T TM221CE••T TM221C••U TM221CE••U
%IWE	20 (for TM221ME16R•)	20 (for TM221ME16T• and TM221ME32TK)	20 (for TM221CE16•)	20 (for TM221CE16•)
%QWM	20 (for TM221ME16R•)	20 (for TM221ME16T• and TM221ME32TK)	20 (for TM221CE16•)	20 (for TM221CE16•)
%IWM	20 (for TM221ME16R•)	20 (for TM221ME16T• and TM221ME32TK)	20 (for TM221CE16•)	20 (for TM221CE16•)
<b>Software objects</b>				
%TM	255	255	255	255
%C	255	255	255	255
%MSG	2	2	1 (for TM221C••R)	1 (for TM221C••T and TM221C••U)
			2 (for TM221CE••R)	2 (for TM221CE••T and TM221CE••U)
%R	4	4	4	4
%DR	8	8	8	8
%SBR	8	8	8	8
%SC	8	8	8	8
%SCH	16	16	16	16
PID	14	14	14	14
<b>Communication objects</b>				
%READ_VAR	16	16	16	16
%WRITE_VAR	16	16	16	16
%WRITE_READ_VAR	16	16	16	16
%SEND_RECV_MSG	16	16	16	16
%SEND_RECV_SMS	1	1	1	1
* for firmware and software version < 1.3				

### Maximum Number of PTO Objects Description

This table provides information about the maximum number of PTO objects supported by the M221 Logic Controller:

Categories/Objects	M221 Logic Controller References		
	TM221M16R• TM221ME16R• TM221C••R TM221CE••R	TM221M16T• TM221ME16T• TM221M32TK TM221ME32TK TM221C••T TM221CE••T TM221C16U TM221CE16U TM221C24U TM221CE24U	TM221C40U TM221CE40U
<b>Motion/Single-axis</b>			
%MC_POWER_PTO	0	86	
%MC_MOVEVEL_PTO			
%MC_MOVEREL_PTO			
%MC_MOVEABS_PTO			
%MC_HOME_PTO			
%MC_SETPOS_PTO			
%MC_STOP_PTO			
%MC_HALT_PTO			
<b>Motion/Motion Task</b>			
%MC_MotionTask_PTO	0	2	4
<b>Administrative</b>			
%MC_READACTVEL_PTO	0	40	
%MC_READACTPOS_PTO			
%MC_READSTS_PTO			
%MC_READMOTIONSTATE_PTO			
%MC_READAXISERROR_PTO			
%MC_RESET_PTO			
%MC_TOUCHPROBE_PTO			
%MC_ABORTTRIGGER_PTO			
%MC_READPAR_PTO			
%MC_WRITEPAR_PTO			

## Section 2.2

### Task Structure

---

#### What Is in This Section?

This section contains the following topics:

Topic	Page
Tasks and Scan Modes	43
Maximum Number of Tasks and Priorities	46

---

## Tasks and Scan Modes

### Overview

The Modicon TM221M Logic Controller supports the following task types:

- Master task
- Periodic task
- Event task

The master tasks can be configured in either of the following scan modes:

- Freewheeling mode
- Periodic mode

For more information, refer to the *Configuring Program Behavior and Tasks* (see *SoMachine Basic, Operating Guide*).

### Tasks

Master tasks are triggered by continuous cyclic scanning or by the software times by specifying the scan period 2...150 ms (default 100 ms) in the periodic mode.

Periodic tasks are triggered by software timers, so are configured by specifying the scan period 2...255 ms (default 255 ms) in the periodic mode.

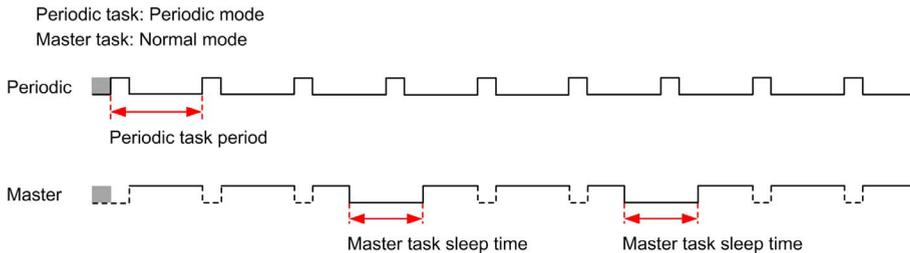
Event tasks are triggered by the physical inputs or the HSC function blocks. These events are associated with embedded digital inputs (%I0.2...%I0.5) (rising, falling or both edges) or with the high speed counters (when the count reaches the high speed counter threshold). You can configure up to two events for each HSC function block, depending on the configuration.

You must configure one priority for each event task. The priority range is 0...7 and the priority 0 has the highest priority.

## Scan Modes

The freewheeling mode is a continuous cyclic scanning mode. In this mode, a new scan starts immediately after the previous scan has completed.

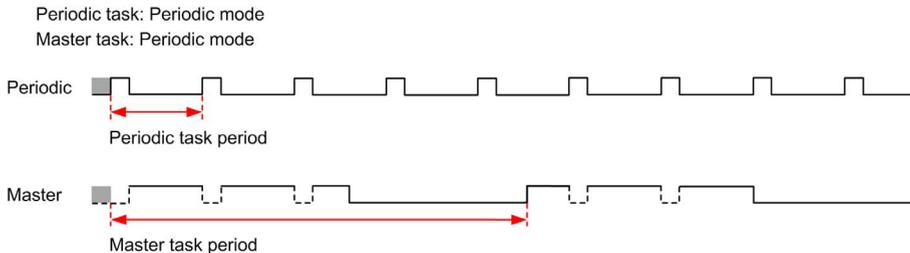
This figure presents the relationship between master tasks and periodic tasks when the master task is in freewheeling mode:



In freewheeling mode, the master task sleep time is at least 30% of the total cycle time with a minimum of 1 millisecond. This percentage may be higher depending on the user application (periodic task scan time, event task scan time, communication interaction, and so on).

In periodic mode, the logic controller waits until the configured scan time has elapsed before starting a new scan. Every scan is therefore the same duration.

This figure presents the relationship between master tasks and periodic tasks when the master task is in periodic mode:

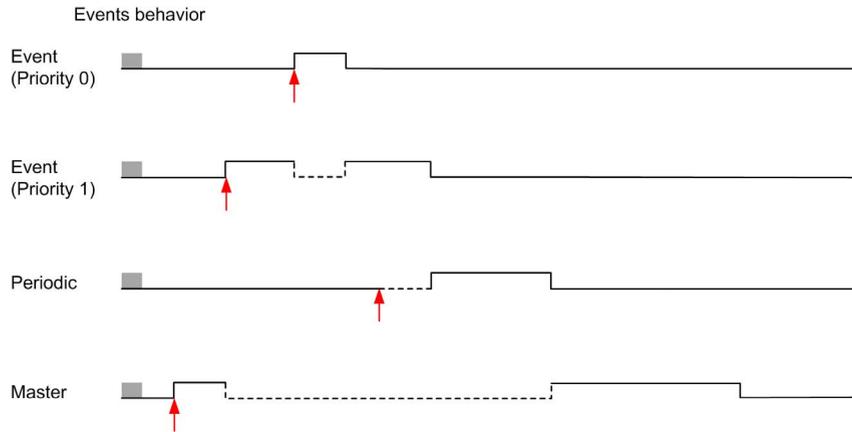


If the processor goes to the `HALTED` state when the master task is configured in freewheeling mode, verify whether the periodic task scan delay time is significant in comparison to the periodic task period. If so, try:

- reconfiguring the master freewheeling task as a cyclic task
- increasing the periodic task period.

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 presents 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.

### Watchdog Timer

You can configure a specific application watchdog timer for the master task and periodic task. If the task execution time exceeds the configured watchdog timer period, the logic controller goes to the `HALTED` state.

A system watchdog timer verifies whether the program is using more than 80% of the processing capacity. In this case, the logic controller goes in the `HALTED` state.

## Maximum Number of Tasks and Priorities

### Description

This table summarizes the task types, available scan modes for each task, scan mode triggering conditions, operator configurable ranges, maximum number of each task, and their execution priorities:

Task Type	Scan Mode	Triggering Condition	Configurable Range	Maximum Number of Tasks	Priority
Master	Freewheeling	Normal	Not applicable	1	Lowest
	Periodic	Software timer	2...150 ms		
Periodic	Periodic	Software timer	2...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	Up to 2 events per %HSC object	4	

---

## Section 2.3

### Controller States and Behaviors

---

#### Introduction

This section provides you with information on controller states, state transitions, and behaviors in response to system events. It begins with a detailed controller state diagram and a description of each state. It then defines the relationship of output states to controller states before explaining the commands and events that result in state transitions. It concludes with information about persistent variables and the effect of SoMachine Basic task programming options on the behavior of your system.

#### What Is in This Section?

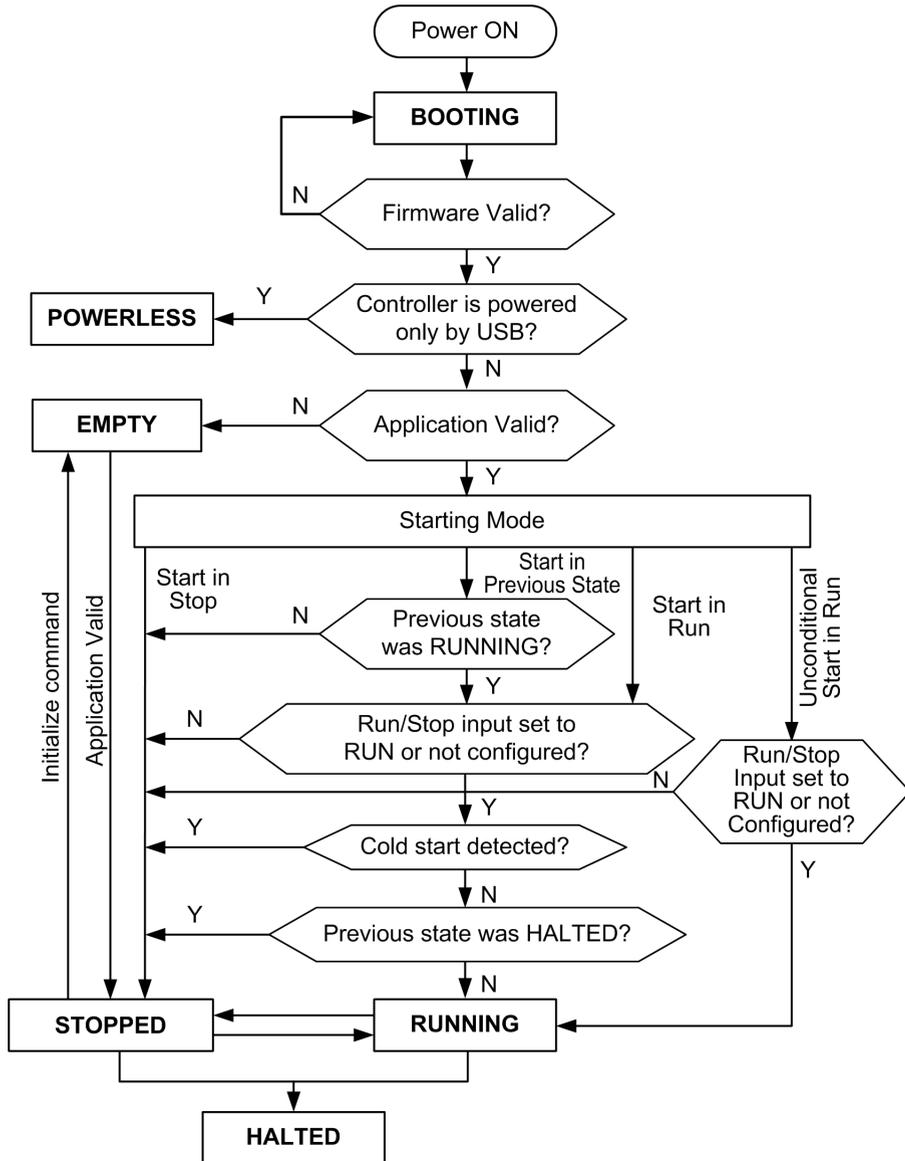
This section contains the following topics:

Topic	Page
Controller States Diagram	48
Controller States Description	49
Controller State Transitions	53
Persistent Variables	56
Output Behavior	57

## Controller States Diagram

### Controller States Diagram

This figure describes the controller states:



## Controller States Description

### Introduction

This section provides a detailed description of the controller states.

#### **WARNING**

##### **UNINTENDED EQUIPMENT OPERATION**

- Never assume that your controller is in a certain controller state before commanding a change of state, configuring your controller options, uploading a program, or modifying the physical configuration of the controller and its connected equipment.
- Before performing any of these operations, consider the effect on all connected equipment.
- Before acting on a controller, always positively confirm the controller state by viewing its LEDs, confirming the condition of the Run/Stop input, checking for the presence of output forcing, and reviewing the controller status information via SoMachine Basic.

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

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.**

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.**

For more information about the Unconditional Start In Run feature, refer to Application Behavior (see *SoMachine Basic, Operating Guide*).

## Controller States Table

This table provides detailed description of the controller operating states:

Controller State	Description	Communication	Application Execution	LED		
				PWR	RUN	ERR
BOOTING	The logic controller does not have a valid firmware. The communication channels are enabled to allow updating of the runtime firmware. It is not possible to login with SoMachine Basic. Outputs are set to initialization values ( <a href="#">see page 57</a> ).	Restricted	No	On	Off	On
EMPTY	This state indicates that there is not a valid application. It is possible to login with SoMachine Basic ( <a href="#">download/animation table</a> ). Inputs are forced to 0. Outputs are set to initialization values ( <a href="#">see page 57</a> ).	Yes	No	On	Off	1 flash
STOPPED	This state indicates that the logic controller has a valid application which is stopped. Inputs are read. Outputs are set to fallback values ( <a href="#">see page 58</a> ), or forced values ( <a href="#">see page 59</a> ) from SoMachine Basic. Status alarm output is set to 0.	Yes	No	On	Flashing	Off
RUNNING	This state indicates that the logic controller is executing the application. Inputs are read by the application tasks. Outputs are written by the application tasks, or from SoMachine Basic in online mode ( <a href="#">animation table</a> , <a href="#">output forcing</a> ( <a href="#">see page 59</a> )). Status alarm output is set to 1.	Yes	Yes	On	On	Off

Controller State	Description	Communication	Application Execution	LED		
				PWR	RUN	ERR
HALTED	<p>This state indicates that the application is stopped because an application or system watchdog timeout error has been detected. <i>(see page 156)</i></p> <p>Objects retain their values, allowing analysis of the cause of the detected error. The tasks are stopped at the last instruction.</p> <p>The communication capabilities are the same as in STOPPED state. Inputs are not read, and keep their last values.</p> <p>Outputs are set to fallback values <i>(see page 58)</i>.</p> <p>Status alarm output is set to 0.</p>	Yes	No	On	Flashing	On
POWERLESS	<p>This state indicates that the logic controller is powered only by the USB cable. This mode can be used to update the firmware (by USB) or to download/upload the user application (by USB).</p> <p>To change the state of the logic controller, connect the main power so that the logic controller boots and reloads the installed components.</p> <p>It is possible to login with SoMachine Basic <i>(download/upload/animation table)</i>.</p> <p>Inputs are forced to 0.</p> <p>Outputs are set to initialization values <i>(see page 57)</i>.</p>	Yes (only USB)	No	Off	Flashing	Off

**NOTE:** The system word %SW6 indicates the logic controller state (EMPTY, STOPPED, RUNNING, HALTED, and POWERLESS).

---

## Controller State Transitions

### Boot Controller

Effect: Command a reboot of the logic controller. For details about power-on sequence, refer to the controller states diagram (*see page 48*).

Methods:

- Power cycle
- Reboot by script
  - The script on an SD card can issue a REBOOT as its last command.

### Application Download

Effect: Download the application into the logic controller memory.

Methods:

- SoMachine Basic online button:
  - Select the **PC to controller (download)** command.
    - Effect: Erase the application in the logic controller and set the logic controller in `EMPTY` state. Download the application into the logic controller memory. If download is successful, a Cold Start is done and the logic controller is set in `STOPPED` state.
- Application file transfer by SD card:
  - Effect: At the next reboot, erase the application in the logic controller and download the application files from the SD card to the controller memory. If download is successful, a Cold Start is done and the controller is set in `STOPPED` state.

### Initialize Controller

Effect: Set the controller in `EMPTY` state, and then, after a Cold Start, in `STOPPED` state.

Methods:

- SoMachine Basic online button:
  - Select the **Initialize controller** command.
- Remote Graphic Display.

## RUN Controller

Effect: Command a transition to the `RUNNING` controller state.

Methods:

- Run/Stop switch on front face:
  - It commands a transition to `RUNNING` state on rising edge.
- Run/Stop input:
  - The input must be configured in the application (Configuring Digital Inputs ([see page 80](#))).
  - It commands a transition to `RUNNING` state on rising edge.
- SoMachine Basic online button:
  - Select the **Run Controller** command.
- Application starting mode setting:
  - **Start in Run**, **Start in Previous State**, or **Unconditional Start in Run**
- Remote Graphic Display.

## STOP Controller

Effect: Command a transition to the `STOPPED` state.

Methods:

- Run/Stop switch on front face:
  - It forces a transition to `STOPPED` state on low level.
- Run/Stop input:
  - The input must be configured in the application (Configuring Digital Inputs ([see page 80](#))).
  - It forces a transition to `STOPPED` state on low level.
- SoMachine Basic online button:
  - Select the **Stop Controller** command.
- Application starting mode setting:
  - **Start in Stop** or **Start in Previous State**.
- **Download** command:
  - It needs the controller to be set in `STOPPED` state (after the download the controller is in `STOPPED` state).
- Remote Graphic Display.

## Error Detected (Transition to `HALTED` State)

Effect: Command a transition to the `HALTED` state.

Reasons for switching to `HALTED` state:

- Application Watchdog timeout (configured by the user)
- System Watchdog timeout (system overrun, over 80% of the processing capacity is used) ([see page 45](#))

## Cold Start

Cold Start is defined to be a power-up with all data initialized to its default values, and program started from the beginning with program variables cleared. Software and hardware settings are initialized.

Cold Start occurs for the following reasons:

- Boot controller without validated application online modification.
- Apply power to a logic controller without a charged backup battery.
- Download application
- Initialize logic controller

### Effects of the Cold Start:

- Initialize the function blocks.
- Clear the user memory.
- Put system objects %S and system words %SW to their initial values.
- Reload parameters from post configuration (changes in the post configuration are applied).
- Restore application from non-volatile memory (unsaved online changes are lost).
- Restart the internal components of the controller.

## Warm Start

The Warm Start resumes running the program, in its previous operating state, with the counters, function blocks, and system words and bits maintained.

## Persistent Variables

### Automatic Save on Power Outage

The controller automatically saves the first 50 memory words (%MW0 to %MW49) in the non-volatile memory following any removal of power. The data is restored to the memory word region during the initialization, even if the controller performs a cold start due to a battery malfunction. These automatically saved persistent variables are reinitialized in case of a new download, initialization command, or system bit %S0 activation (refer to System Bits ([see page 180](#))).

### Save by User Request

You can save up to 1000 memory words (%MW50 up to %MW1049) in the non-volatile memory. The number of memory words saved is specified in the system word %SW148 (refer to System Words ([see page 188](#))).

To perform the save operation, set system bit %S93 to 1 (refer to System Bits ([see page 180](#))). The non-volatile memory region containing %MW50 to %MW1049 is erased at the end of the master task cycle. The operation occupies the controller for approximately 40 ms when the controller is in STOPPED state. However, when the controller is in RUNNING state, it can take several seconds, depending on the load on the controller processor. This also impacts communications. The system bit %S92 (refer to System Bits ([see page 180](#))) is set to 1 to signal the end of the save operation.

### Restore by User Request

To restore “Save by User Request” memory words, set system bit %S94 to 1 (refer to System Bits ([see page 180](#))). The restore operation is performed at the end of the master task cycle. The number of words restored is written to the system word %SW148 (refer to System Words ([see page 188](#))).

## Output Behavior

### Introduction

The controller defines output behavior in response to commands and system events in a way that allows for greater flexibility. An understanding of this behavior is necessary before discussing the commands and events that affect controller states.

The possible output behaviors and the controller states to which they apply are:

- Managed by application
- Initialization values
- Fallback Management (see *SoMachine Basic, Operating Guide*)
  - Maintain values
  - Fallback values
- Output forcing

### Managed by Application

Your application manages outputs normally. This applies in the `RUNNING` state.

### Hardware Initialization Values

This output state applies in the `BOOTING`, `EMPTY`, and `POWERLESS` states.

In the initialization state, the outputs assume the following values:

- For embedded outputs:
  - Fast transistor output: 0 Vdc
  - Regular transistor output: 0 Vdc
  - Relay output: Open
  - Expert I/O functions (HSC, PLS, PWM, PTO and FREQGEN): 0 Vdc
- For expansion module outputs:
  - Regular transistor output: 0 Vdc
  - Relay output: Open

### Software Initialization Values

This output state applies when downloading or when resetting the application. It applies at the end of the download or at the end of a warm start or cold start.

Input objects `%I` and `%IW`, and network objects `%QWE` and `%QWM`, are set to 0. Output objects `%Q` and `%QW`, and network objects `%IWE` and `%IWM`, are set according to the fallback strategy.

### Fallback Management

The objective of the fallback behavior is to control the outputs when the controller leaves the `RUNNING` state.

Fallback values are applied on the transition from `RUNNING` to `STOPPED` or `HALTED` states, except for special cases described below.

## Fallback Behavior Configuration

Fallback behavior is configured on the **Programming** tab, **Tasks** → **Behavior** window:

- When **Maintain values** checkbox is not selected, on a fallback occurrence, output values take the values configured in **Fallback value**.
- When **Maintain values** checkbox is selected, outputs keep their values on a fallback occurrence, except for outputs configured in pulse generator (PWM, PLS, PTO, FREQGEN) or reflex functions.

## Fallback Execution

On a fallback occurrence:

- If **Maintain values** is not selected, the outputs take the values configured in **Fallback value**.
- If **Maintain values** is selected, the outputs keep their values.

Special cases:

- Alarm output: The fallback is never applied in the alarm output. If alarm output is configured, its fallback value is forced to 0.
- Pulse generators (PLS, PWM, PTO, FREQGEN) and reflex outputs:
  - If **Maintain values** is not selected, the outputs take the values configured in **Fallback value**.
  - If **Maintain values** is selected, the outputs are set to 0.

### NOTE:

- After a download, the outputs are set to 0 (default values).
- In `EMPTY` state, the outputs are set to 0.
- As the data image reflects the physical values, fallback values are also applied to the data image.

## Fallback Values

This output state applies in the `STOPPED` and `HALTED` states.

During fallback, the outputs assume the following values:

- For embedded outputs:
  - Fast transistor output: according to fallback setting
  - Regular transistor output: according to fallback setting
  - Relay output: according to fallback setting
  - Expert I/O functions (HSC, PLS, PWM, PTO, and FREQGEN):
    - Maintain value: 0 Vdc
- For expansion module outputs:
  - Regular transistor output: according to fallback setting
  - Relay output: according to fallback setting

## Output Forcing

The controller allows you to force the state of selected outputs to a defined value for the purposes of system testing, commissioning, and maintenance.

You are only able to force the value of an output while your controller is connected to SoMachine Basic.

To do so, use the **Force** command in an animation table.

Output forcing overrides all other commands to an output irrespective of the task programming that is being executed.

The forcing is not released by online change or logout of SoMachine Basic.

The forcing is automatically released by Cold Start ([see page 55](#)) and Download application ([see page 53](#)) command.

The forcing does not apply for expert I/O functions (HSC, PLS, PWM and PTO).

### **WARNING**

#### **UNINTENDED EQUIPMENT OPERATION**

- You must have a thorough understanding of how forcing will affect the outputs relative to the tasks being executed.
- Do not attempt to force I/O that is contained in tasks that you are not certain will be executed in a timely manner, unless your intent is for the forcing to take affect at the next execution of the task whenever that may be.
- If you force an output and there is no apparent affect on the physical output, do not exit SoMachine Basic without removing the forcing.

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

## Output Rearming

In the case of a short-circuit or current overload, the common group of outputs automatically enters into thermal protection mode (all outputs in the group are set to 0), and are then periodically rearmed (each second) to test the connection state. However, you must be aware of the effect of this rearming on the machine or process being controlled.

### **WARNING**

#### **UNINTENDED MACHINE START-UP**

Inhibit the automatic rearming of outputs if this feature is an undesirable behavior for your machine or process.

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

**NOTE:** Only the short-circuit between an output set to `TRUE` and 0 V is detected. The short-circuit between an output set to `FALSE` and 24 V is not detected.

If necessary, you can use system bits and words to both detect that a short circuit or overload has occurred and on which cluster of outputs it has occurred. System bit `%S10` can be used to detect within your program that an output error has occurred. You can then use the system word `%SW139` to determine programmatically which cluster of outputs a short circuit or overload has occurred.

The automatic rearming feature can be disabled by setting the system bit `%S49` to 0 (`%S49` is set to 0 by default).

---

## Section 2.4

### Post Configuration

---

#### Introduction

This section describes how to manage and configure the post configuration file of the Modicon M221 Logic Controller.

#### What Is in This Section?

This section contains the following topics:

Topic	Page
Post Configuration	62
Post Configuration File Management	63

## Post Configuration

### Introduction

Post configuration is an option that allows you to modify some parameters of the application without changing the application. Post configuration parameters are defined in a file called **Machine.cfg**, which is stored in the controller.

By default, all communication parameters are set in the configuration of the application. However, under certain conditions, some or all of these parameters can be modified automatically via the mechanism Post Configuration. One or more communication parameters can be specified in the post configuration file, and those parameters can override the parameters specified by the configuration. For example, one parameter may be stored in the post configuration file to change the Ethernet IP address of the controller while leaving the other Ethernet parameters, such as the gateway address, unchanged.

### Parameters

The post configuration file allows you to modify network parameters.

Ethernet parameters:

- Address configuration mode
- IP address
- Subnet mask
- Gateway address
- Device name

Serial line parameters, for each serial line in the application (embedded port or TMC2SL1 cartridge):

- Physical medium
- Baud rate
- Parity
- Data bits
- Stop bit
- Modbus address
- Polarization (for RS-485)

### Operating Mode

The post configuration file is read and applied:

- after a Warm Start ([see page 55](#))
- after a Cold Start ([see page 55](#))
- after a reboot ([see page 53](#))
- after an application download ([see page 53](#))
- after an Ethernet reconfiguration caused by an Ethernet cable reconnection (exclusively for the Ethernet part of the post configuration file ([see page 121](#)))

For further details on controller states and transitions, refer to Controller States and Behaviors ([see page 47](#)).

## Post Configuration File Management

### Introduction

The post configuration file can be transferred, modified, or deleted with an SD card. Refer to Post Configuration Management ([see page 154](#)).

**NOTE:** A post configuration file example is available in the directory `Firmwares & PostConfiguration\PostConfiguration\add_change\usr\cfg` of the SoMachine Basic installation directory.

### Post Configuration File Format

A valid configuration must use the following format:

- The character '#' means beginning of comment, everything after this sign until the end of the line is ignored. Comments are not saved in the post configuration area of the M221 Logic Controller.
- Rule is `channel.parameter=value` (no space around the '=' sign).
- Channel and parameter are case-sensitive.
- Allowed channel, parameter, and values are in the following table.

Channel	Parameter	Description	Value
ETH	IPMODE	Address configuration mode	0 = Fixed 1 = BOOTP 2 = DHCP
	IP	IP address	Dotted decimal string
	MASK	Subnet mask	Dotted decimal string
	GATEWAY	Gateway address	Dotted decimal string
	NETWORKNAME	Device name on the network	ASCII string (maximum 16 characters)
SL1 SL2	HW	Physical medium	0 = RS-232 1 = RS-485
	BAUDS	Data transmission rate	1200, 2400, 4800, 9600, 19200, 38400, 57600 or 115200
	PARITY	Parity for error detection	0 = None 1 = Odd 2 = Even
	DATAFORMAT	Data format	7 or 8
	STOPBIT	Stop bit	1 or 2
	MODBUSADDR	Modbus address	1...247
	POLARIZATION	Polarization (for cartridges only)	0 = No 1 = Yes

**NOTE:**

When using a post configuration file for Ethernet configuration, it is not mandatory to specify all the parameters:

- If the M221 Logic Controller is configured (by the user application) in DHCP or BOOTP mode, the network parameters IP (IP address), MASK (subnet mask) and GATEWAY (gateway address) are not configured in the file.
- If a parameter is not configured in the post configuration file, the M221 Logic Controller uses the value configured in the user application (see Ethernet configuration ([see page 120](#))).
- If the M221 Logic Controller is configured in DHCP or BOOTP mode by the user application and if fixed IP mode (IPMODE=0) is configured in the post configuration file, configure the network parameters (IP (IP address), MASK (subnet mask) and GATEWAY (gateway address)) as they are not configured by the user application. Otherwise, the M221 Logic Controller starts with the default Ethernet configuration.

### Post Configuration File Transfer

After creating and modifying your post configuration file, it must be transferred to the logic controller. The transfer is performed by copying the post configuration file to an SD card with a script.

Refer to Adding or Changing a Post Configuration ([see page 154](#)).

### Modifying a Post Configuration File

Use a text editor to modify the post configuration file on the PC.

**NOTE:** Do not change the text file encoding. The default encoding is ANSI.

**NOTE:** The Ethernet parameters of the post configuration file can be modified with SoMachine Basic. For more information, refer to Connecting to a Logic Controller ([see SoMachine Basic, Operating Guide](#)).

### Deleting the Post Configuration File

Refer to Removing a Post Configuration File ([see page 155](#)).

**NOTE:** The parameters defined in the application will be used instead of the corresponding parameters defined in the post configuration file.

---

# Part II

## Configuring the M221 Logic Controller

---

### Overview

This part provides information about how to configure the M221 Logic Controller references.

### What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
3	How to Configure a Controller	67
4	Embedded Input/Output Configuration	79
5	I/O Bus Configuration	111
6	Embedded Communication Configuration	119
7	SD Card	143



---

# Chapter 3

## How to Configure a Controller

---

### Overview

This chapter describes how to build a configuration in SoMachine Basic and configure the M221 Logic Controller.

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Building a Configuration	68
Optional I/O Expansion Modules	73
Configuring the M221 Logic Controller	77
Updating Firmware using Executive Loader Wizard	78

## Building a Configuration

### Introduction

Configure a controller by building a configuration in SoMachine Basic. To build a configuration, first create a new project or open an existing project.

Refer to *SoMachine Basic Operating Guide* for information on how to:

- create or open an existing project
- replace the default logic controller
- add an expansion module to the logic controller
- add a cartridge to the logic controller
- save the project.

Some general information about the SoMachine Basic user interface is provided below.

### Start Page

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

### SoMachine Basic Window

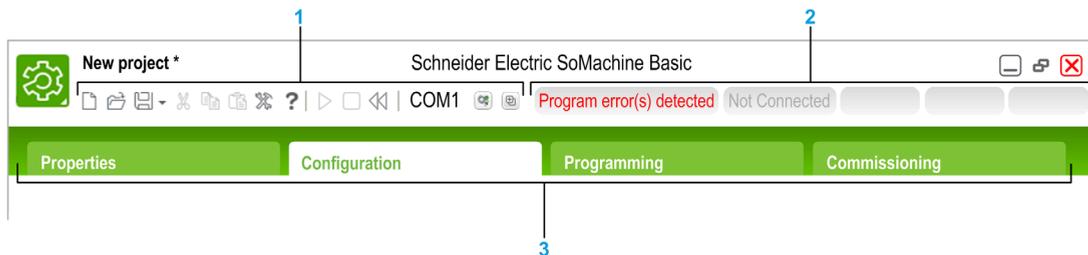
Once you have selected a project to work with, SoMachine Basic displays the main window.

At the top of the main window, a toolbar (see *SoMachine Basic, Operating Guide*) contains icons that allow you to perform common tasks, including returning to the start page window.

Next to the toolbar, the status bar (see *SoMachine Basic, Operating Guide*) displays informational messages about the state of the connection to the logic controller.

Below the toolbar and the status bar, 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 the module tab.

This figure presents the toolbar, status bar, and the module tabs in the main window:



- 1 Toolbar
- 2 Status bar
- 3 Module tabs

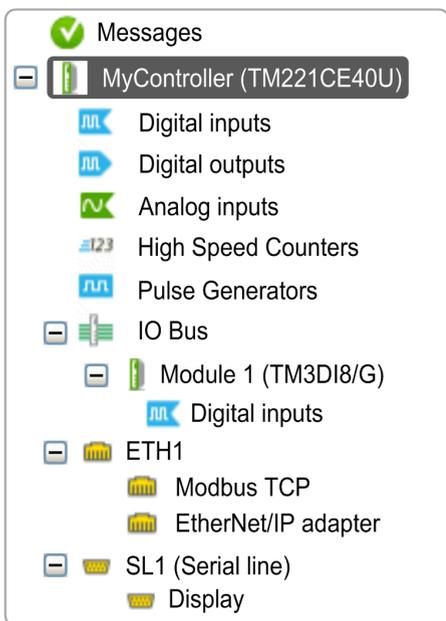
Item	Description
Toolbar	Provides easy access to commonly used functions. For more information, refer to the Toolbar (see <i>SoMachine Basic, Operating Guide</i> ).
Status bar	Displays status and information messages on the system status. For more information, refer to the Status bar (see <i>SoMachine Basic, Operating Guide</i> ).
Module tabs	<p>To develop an application, work your way through the module tabs from left to right:</p> <ul style="list-style-type: none"> <li>● <b>Properties</b> Set up the project properties.</li> <li>● <b>Configuration</b> Replicate and configure the hardware configuration of the logic controller and associated expansion modules.</li> <li>● <b>Programming</b> Develop the program in one of the supported programming languages.</li> <li>● <b>Display</b> Build an operator interface for a Remote Graphic Display device. Refer to the TMH2GDB Remote Graphic Display User Guide for details.</li> <li>● <b>Commissioning</b> Manage the connection between SoMachine Basic and the logic controller, upload/download applications, test, and commission the application.</li> </ul>

### Hardware Tree

The hardware tree is displayed on left-hand side in the **Configuration** window. It shows a structured view of the hardware configuration. When you add a controller, an expansion module, or a cartridge to the project, several nodes are automatically added to the hardware tree.

**NOTE:** The nodes in the hardware tree are specific to the controller and the hardware configuration. These nodes depend on the I/O functions that the controller, expansion modules, and cartridges provide.

This figure presents the hardware tree of the controller configuration:



Item	Description
<b>Digital inputs</b>	Use to configure the embedded digital inputs of the logic controller.
<b>Digital outputs</b>	Use to configure the embedded digital outputs of the logic controller.
<b>Analog inputs</b>	Use to configure the embedded analog inputs of the logic controller.
<b>High Speed Counters</b>	Use to configure the embedded high speed counting functions (HSC).
<b>Pulse Generators</b>	Use to configure the embedded pulse generator functions (PLS/PWM/PTO/FREQGEN).
<b>IO Bus</b>	Use to configure the expansion modules and cartridges connected to the logic controller.
<b>ETH1</b>	Use to configure the embedded Ethernet communications.
<b>Modbus TCP</b>	Use to configure the Modbus TCP protocol for Ethernet communications.
<b>EtherNet/IP adapter</b>	Use to configure the EtherNet/IP adapter for Ethernet communications.
<b>SLn (Serial line)</b>	Use to configure the embedded serial line or the serial line added using a cartridge.
n Serial line number (1 or 2, controller-specific).	

## Editor

The editor area is displayed in center of the **Configuration** window. It displays the graphical representation of hardware configuration of the devices. The hardware configuration in a project can be:

- only a controller
- a controller with cartridges
- a controller with expansion modules
- a controller with cartridges and expansion modules.

The editor area displays:

- a short description about the device when you click the device image or when you click the device node in the hardware tree.
- configuration properties of the item selected in the hardware tree.

If you add an expansion module to the configuration, the expansion module appears at the right-hand side of the controller or the previously added expansion module. Cartridges are added on the controller in the cartridge slot.

When configuring a controller, a cartridge, or an expansion module, the configuration properties of the node selected in the hardware tree are displayed below the graphical configuration. These properties allow you to configure the device.

This figure presents the configuration of a controller with an expansion module (the controller is selected):



### Catalog

The catalog area is displayed on right-hand side in the **Configuration** window. It displays the complete range of the logic controllers, expansion modules, and cartridges that can be configured using SoMachine Basic. It also provides a short description of the selected device.

You can drag-and-drop the objects from the catalog area to the editor area. You can also replace the existing controller by a different controller with simple drag-and-drop from the catalog.

This figure presents the catalog of the logic controllers and the expansion modules:

▼ M221 Logic Controllers

Reference	Type	Comm. Ports	Digital Input	Digital Output
TM221CE40R	Compact Vac	1 SL + 1 ETH	24	16 relays
TM221CE40T	Compact 24Vdc	1 SL + 1 ETH	24	16 transistors
TM221M16R/G	Modular 24Vdc	2 SL	8	8 relays
TM221M16T/G	Modular 24Vdc	2 SL	8	8 transistors
TM221M32TK	Modular 24Vdc	2 SL	16	16 transistors
TM221ME16R/G	Modular 24Vdc	1 SL + 1 ETH	8	8 relays
TM221ME16T/G	Modular 24Vdc	1 SL + 1 ETH	8	8 transistors
TM221ME32TK	Modular 24Vdc	1 SL + 1 ETH	16	16 transistors

- > TM3 Digital I/O Modules
- > TM3 Analog I/O Modules
- > TM2 Digital I/O Modules
- > TM2 Analog I/O Modules
- > TM3 Expert I/O Modules
- > M221 Cartridges

**Device description**

TM221M16R (screw), TM221M16RG (spring)  
 8 digital inputs, 8 relay outputs (2 A), 2 analog inputs, 2 serial line ports, 24 Vdc modular controller with removable terminal blocks.

<b>5 V</b>	<b>24 V</b>
520 mA	432 mA



## Optional I/O Expansion Modules

### Presentation

I/O expansion modules can be marked as optional in the configuration. The **Optional module** feature provides a more flexible configuration by the acceptance of the definition of modules that are not physically attached to the logic controller. Therefore, a single application can support multiple physical configurations of I/O expansion modules, allowing a greater degree of scalability without the necessity of maintaining multiple application files for the same application.

Without the **Optional module** feature, when the logic controller starts up the I/O expansion bus (following a power cycle, application download or initialization command), it compares the configuration defined in the application with the physical I/O modules attached to the I/O bus. Among other diagnostics made, if the logic controller determines that there are I/O modules defined in the configuration that are not physically present on the I/O bus, an error is detected and the I/O bus does not start.

With the **Optional module** feature, the logic controller ignores the absent I/O expansion modules that you have marked as optional, which then allows the logic controller to start the I/O expansion bus.

The logic controller starts the I/O expansion bus at configuration time (following a power cycle, application download, or initialization command) even if optional expansion modules are not physically connected to the logic controller.

The following module types can be marked as optional:

- TM3 I/O expansion modules
- TM2 I/O expansion modules

**NOTE:** TM3 Transmitter/Receiver modules (TM3XTRA1 and the TM3XREC1) and TMC2 cartridges cannot be marked as optional.

The application must be configured with a functional level of at least **Level 3.2** for modules marked as optional to be recognized as such by the logic controller.

You must be fully aware of the implications and impacts of marking I/O modules as optional in your application, both when those modules are physically absent and present when running your machine or process. Be sure to include this feature in your risk analysis.

### **WARNING**

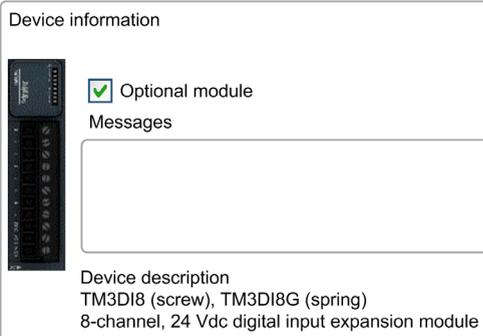
#### **UNINTENDED EQUIPMENT OPERATION**

Include in your risk analysis each of the variations of I/O configurations that can be realized marking I/O expansion modules as optional, and in particular the establishment of TM3 Safety modules (TM3S...) as optional I/O modules, and make a determination whether it is acceptable as it relates to your application.

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

### Marking an I/O Expansion Module as Optional in Offline Mode

To add a module and mark it as optional in the configuration:

Step	Action
1	Drag-and-drop the I/O expansion module from the catalog to the editor.
2	<p>In the <b>Device information</b> area, select the <b>Optional module</b> check box:</p> <div style="border: 1px solid black; padding: 5px;"> <p>Device information</p>  <p>Device description                      TM3DI8 (screw), TM3DI8G (spring)                      8-channel, 24 Vdc digital input expansion module</p> </div>

To mark an existing I/O expansion module as optional in the configuration:

Step	Action
1	Select the I/O expansion module in the editor.
2	In the <b>Device information</b> area, select the <b>Optional module</b> check box.

## Optional I/O Expansion Modules in Online Mode

SoMachine Basic operates in online mode when a physical connection to a logic controller has been established.

When in SoMachine Basic online mode, the modification of the **Optional module** feature is disabled. You can visualize the downloaded configuration in the application:

- An I/O expansion module represented in yellow is marked as optional and not physically connected to the logic controller at start-up. An information message to that effect is displayed in the **Device information** area.
- An I/O expansion module represented in red is not marked as optional and not detected at start-up. An information message to that effect is displayed in the **Device information** area.

The selection of the **Optional module** feature is used by the logic controller to start the I/O bus. The following system words are updated to indicate the status of the physical I/O bus configuration:

System Word	Comment
%SW118 Logic controller status word	Bits 13 and 14 are pertinent to the I/O module status relative to the I/O bus. Bit 13, if FALSE, indicates that there are mandatory modules as defined by the I/O expansion bus configuration that are absent or otherwise inoperative when the logic controller attempts to start the I/O expansion bus. In this case, the I/O bus does not start. Bit 14, if FALSE, indicates that one or more modules have ceased communication with the logic controller after the I/O expansion bus is started. This is the case whether an I/O expansion module is defined as mandatory or as an optional module but present at start-up.
%SW119 I/O expansion module configuration	Each bit, starting with bit 1 (bit 0 is reserved), is dedicated to a configured I/O expansion module and indicates whether the module is optional (TRUE) or mandatory (FALSE) when the controller attempts to start the I/O bus.
%SW120 I/O expansion module status	Each bit, starting with bit 1 (bit 0 is reserved), is dedicated to a configured I/O expansion module and indicates the status of the module. When the logic controller attempts to start the I/O bus, if the value of %SW120 is non-zero (indicating that an error is detected for at least one of the modules), the I/O expansion bus does not start unless the corresponding bit in %SW119 is set to TRUE (indicating the module is marked as an optional module). When the I/O bus is started, if the value of %SW120 is modified by the system, it indicates that an error is detected on one or more I/O expansion modules (regardless of the <b>Optional module</b> feature).

For more information, refer to System Words ([see page 188](#)).

## Shared Internal ID Codes

Logic controllers identify expansion modules by a simple internal ID code. This ID code is not specific to each reference, but identifies the structure of the expansion module. Therefore, different references can share the same ID code.

You cannot have two modules with the same internal ID code declared as optional without at least one mandatory module placed between them.

This table groups the module references sharing the same internal ID code:

<b>Modules sharing the same internal ID code</b>
TM2DDI16DT, TM2DDI16DK
TM2DRA16RT, TM2DDO16UK, TM2DDO16TK
TM2DDI8DT, TM2DAI8DT
TM2DRA8RT, TM2DDO8UT, TM2DDO8TT
TM2DDO32TK, TM2DDO32UK
TM3DI16K, TM3DI16/G
TM3DQ16R/G, TM3DQ16T/G, TM3DQ16TK, TM3DQ16U, TM3DQ16UG, TM3DQ16UK
TM3DQ32TK, TM3DQ32UK
TM3DI8/G, TM3DI8A
TM3DQ8R/G, TM3DQ8T/G, TM3DQ8U, TM3DQ8UG
TM3DM8R/G
TM3DM24R/G
TM3SAK6R/G
TM3SAF5R/G
TM3SAC5R/G
TM3SAFL5R/G
TM3AI2H/G
TM3AI4/G
TM3AI8/G
TM3AQ2/G
TM3AQ4/G
TM3AM6/G
TM3TM3/G
TM3TI4/G
TM3TI8T/G

## Configuring the M221 Logic Controller

### Controller Configuration

Controller configuration depends on the number and type of embedded input/outputs, I/O objects, and communication ports.

Use the **Configuration** tab to configure the properties of your controller and the expansion modules. Select a node in the hardware tree to configure the properties of the controller.

This table shows the available configurations of the M221 Logic Controller:

Reference	Digital Input	Digital Output	Analog Input	High Speed Counter	Pulse Generator	Ethernet	Serial Line
TM221M16R• TM221C••R TM221C••U	X	X	X	X			X
TM221ME16R• TM221CE••R TM221CE••U	X	X	X	X		X	X
TM221M16T• TM221M32TK TM221C••T	X	X	X	X	X		X
TM221ME16T• TM221ME32TK TM221CE••T TM221CE••U	X	X	X	X	X	X	X

**X** Available for configuration in SoMachine Basic. For information on how to configure:

- Digital inputs, refer to Configuring Digital Inputs ([see page 80](#))
- Digital outputs, refer to Configuring Digital Outputs ([see page 84](#))
- Analog inputs, refer to Configuring Analog Inputs ([see page 86](#))
- High speed counters, refer to Configuring High Speed Counters ([see page 89](#))
- Pulse generators, refer to Configuring Pulse Generators ([see page 99](#))
- Ethernet, refer to Configuring Ethernet ([see page 120](#))
- Serial lines, refer to Configuring Serial Line ([see page 133](#)).

## Updating Firmware using Executive Loader Wizard

### Overview

You can update the firmware of the controller using the Executive Loader wizard.

Refer to Controller States and Behavior ([see page 47](#)) for information concerning the state of the firmware in your controller.

### Updating the Firmware of the Controller

To launch the **ExecLoader** wizard, follow these steps:

Step	Action
1	Close all Windows applications, including virtual machines.
2	Click <b>Start</b> → <b>Programs</b> → <b>Schneider Electric</b> → <b>SoMachine Basic</b> → <b>SoMachine Basic Firmware Update</b> or run the <i>ExecLoaderWizard.exe</i> from <i>SoMachine Basic installation folder\Execloader</i> folder.

---

# Chapter 4

## Embedded Input/Output Configuration

---

### Overview

This chapter describes how to configure the embedded I/O objects of the M221 Logic Controller.

The number of embedded inputs and outputs depends on the controller reference. For more information, refer to the tables for:

- TM221C Logic Controller (*see page 22*)
- TM221M Logic Controller (*see page 27*)

### What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
4.1	Digital Input Configuration	80
4.2	Digital Output Configuration	84
4.3	Analog Input Configuration	86
4.4	High Speed Counter Configuration	88
4.5	Pulse Generator Configuration	98

# Section 4.1

## Digital Input Configuration

### Configuring Digital Inputs

#### Introduction

By default, all digital inputs are used as regular inputs. Some of the digital inputs are fast and can be used by configuring the high speed counters (*see page 89*) while other inputs can be configured as event sources.

#### Digital Inputs Configuration

This table describes how to configure the digital inputs:

Step	Action																																																																																																			
1	<p>Click the <b>Digital inputs</b> node in the hardware tree to display the digital input properties. This figure shows the properties of the digital inputs in the editor area:</p> <div data-bbox="290 760 1141 1224" style="border: 1px solid gray; padding: 10px;"> <p><b>Digital inputs</b></p> <table border="1"> <thead> <tr> <th>Used</th> <th>Address</th> <th>Symbol</th> <th>Used by</th> <th>Filtering</th> <th>Latch</th> <th>Run/Stop</th> <th>Event</th> <th>Priority</th> <th>Subroutine</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>%I0.0</td> <td></td> <td>Filtering</td> <td>3 ms</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Used</td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%I0.1</td> <td></td> <td>Filtering</td> <td>3 ms</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Used</td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%I0.2</td> <td></td> <td>Filtering</td> <td>3 ms</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Used</td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%I0.3</td> <td></td> <td>Filtering</td> <td>3 ms</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Used</td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%I0.4</td> <td></td> <td>Filtering</td> <td>3 ms</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Used</td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%I0.5</td> <td></td> <td>Filtering</td> <td>3 ms</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Used</td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%I0.6</td> <td></td> <td>Filtering</td> <td>3 ms</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Used</td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%I0.7</td> <td></td> <td>Filtering</td> <td>3 ms</td> <td><input type="checkbox"/></td> <td><input type="checkbox"/></td> <td>Not Used</td> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p style="text-align: right;"> <input type="button" value="Apply"/> <input type="button" value="Cancel"/> </p> </div>	Used	Address	Symbol	Used by	Filtering	Latch	Run/Stop	Event	Priority	Subroutine	Comment	<input type="checkbox"/>	%I0.0		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used				<input type="checkbox"/>	%I0.1		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used				<input type="checkbox"/>	%I0.2		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used				<input type="checkbox"/>	%I0.3		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used				<input type="checkbox"/>	%I0.4		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used				<input type="checkbox"/>	%I0.5		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used				<input type="checkbox"/>	%I0.6		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used				<input type="checkbox"/>	%I0.7		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used			
Used	Address	Symbol	Used by	Filtering	Latch	Run/Stop	Event	Priority	Subroutine	Comment																																																																																										
<input type="checkbox"/>	%I0.0		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used																																																																																													
<input type="checkbox"/>	%I0.1		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used																																																																																													
<input type="checkbox"/>	%I0.2		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used																																																																																													
<input type="checkbox"/>	%I0.3		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used																																																																																													
<input type="checkbox"/>	%I0.4		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used																																																																																													
<input type="checkbox"/>	%I0.5		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used																																																																																													
<input type="checkbox"/>	%I0.6		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used																																																																																													
<input type="checkbox"/>	%I0.7		Filtering	3 ms	<input type="checkbox"/>	<input type="checkbox"/>	Not Used																																																																																													
2	<p>Edit the properties to configure the digital inputs. For detailed information on the digital input configuration parameters, refer to the table below.</p>																																																																																																			

This table describes each parameter of the digital input configuration:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	True/False	False	Indicates whether the input channel is being used in a program or not.
<b>Address</b>	No	%I0.x	–	Displays the address of the digital input on the controller, where x represents the channel number. If the controller has 8 digital input channels, x varies from 0...7. If the controller has 16 digital input channels, x varies from 0...15. For example, %I0.2 is the third digital input channel of the logic controller.
<b>Symbol</b>	Yes	–	–	Allows you to specify a symbol to associate with the digital input object. Double-click in the <b>Symbol</b> column, type the name of the symbol and press <b>Enter</b> .
<b>Used by</b>	No	any	<b>Filtering</b>	Displays the name of the component that uses the input channel. For example, if the input channel is used by a subroutine, this field displays <b>User logic</b> . The possible values in this field are: <ul style="list-style-type: none"> <li>● <b>User logic</b></li> <li>● <b>Filtering</b></li> <li>● <b>Latch</b></li> <li>● <b>Run/Stop</b></li> <li>● <b>Event</b></li> <li>● <b>%HSCx</b> where x is the high speed counter instance on the controller</li> <li>● <b>%FCy</b> where y is the fast counter instance on the controller</li> </ul> If an input is being used by more than one operation, all values, separated by commas, are displayed in this field.
<b>Filtering</b>	Yes	<b>No Filter</b> <b>3 ms</b> <b>12 ms</b>	<b>3 ms</b>	Allows you to select the noise filter duration for the input channel. Using a filter for the digital inputs reduces the noise on the controller input. If you select filter for an input, you cannot configure that input for: <ul style="list-style-type: none"> <li>● <b>Latch</b></li> <li>● <b>Event</b></li> </ul>

Parameter	Editable	Value	Default Value	Description
<b>Latch</b>	Yes	True/False	False	<p>Allows you to enable or disable latching for the inputs configured as events (%IO.2...%IO.5).</p> <p>By default, this option is disabled due to default value of <b>Filtering</b>. Set the <b>Filtering</b> to <b>No Filter</b> to enable the <b>Latch</b> option.</p> <p>Latching enables pulses with a duration shorter than the controller scan time to be memorized.</p> <p>When a pulse duration is shorter than a scan time and has a value greater than or equal to 1 ms, the controller latches the pulse, which is then updated in the next scan.</p> <p>If you enable <b>Latch</b> for an input, you cannot configure that input for:</p> <ul style="list-style-type: none"> <li>● <b>Filtering</b></li> <li>● <b>Run/Stop</b></li> <li>● <b>Event</b></li> </ul>
<b>Run/Stop</b>	Yes	True/False	False	<p>Allows you to configure 1 digital input as an additional Run/Stop switch.</p> <p>If you configure a digital input as Run/Stop switch, you cannot use the input in any other function block (for example, high speed counter function block, fast counter function block, and so on).</p> <p>If you enable <b>Run/Stop</b> for an input, you cannot configure that input for:</p> <ul style="list-style-type: none"> <li>● <b>Latch</b></li> <li>● <b>Event</b></li> </ul>
<b>Event</b>	Yes	<b>Not Used</b> <b>Falling Edge</b> <b>Rising Edge</b> <b>Both edges</b>	<b>Not Used</b>	<p>Allows you to select an event that triggers the inputs %IO.2...%IO.5.</p> <p>By default, this option is disabled due to default value of <b>Filtering</b>. Set the <b>Filtering</b> to <b>No Filter</b> to enable the <b>Event</b> option.</p> <p>If you select an event from the drop-down list (other than <b>Not Used</b>), the <b>Priority</b> parameter enables for editing to set the priority of the event.</p>
<b>Priority</b>	Yes	0...7	7	<p>Allows you to set the priority of the triggering event for the inputs %IO.2...%IO.5.</p> <p>You can set the priority of each event using the <b>Priority</b> parameter that is editable only for the inputs configured as event.</p> <p>Assign each configured event a different priority: if 2 events have same priority, a detected error message appears in the window.</p>

Parameter	Editable	Value	Default Value	Description
<b>Subroutine</b>	No	<i>any</i>	<i>empty</i>	Displays the number of the subroutine associated with an input configured as an event.
<b>Comment</b>	Yes	–	–	Allows you to specify a comment to associate with the digital input object. Double-click in the <b>Comment</b> column, type the comment and press <b>Enter</b> .

Additional configuration details are displayed in the **Programming** tab. For more information, refer to Digital Inputs (%) ([see page 164](#)).

# Section 4.2

## Digital Output Configuration

### Configuring Digital Outputs

#### Introduction

By default, all digital outputs are used as regular outputs. For controllers equipped with transistor outputs, 2 outputs are fast transistor outputs and can be used by configuring the pulse generators (see page 99).

#### Digital Outputs Configuration

This table describes how to configure the digital outputs:

Step	Action																																																															
1	<p>Click the <b>Digital outputs</b> node in the hardware tree to display the digital output properties. This figure shows the properties of the digital outputs in the editor area:</p> <div data-bbox="290 760 1136 1221" style="border: 1px solid gray; padding: 10px;"> <p><b>Digital outputs</b></p> <table border="1"> <thead> <tr> <th>Used</th> <th>Address</th> <th>Symbol</th> <th>Used by</th> <th>Status Alarm</th> <th>Fallback value</th> <th>Comment</th> </tr> </thead> <tbody> <tr><td><input type="checkbox"/></td><td>%Q0.0</td><td></td><td></td><td><input type="checkbox"/></td><td>0</td><td></td></tr> <tr><td><input type="checkbox"/></td><td>%Q0.1</td><td></td><td></td><td><input type="checkbox"/></td><td>0</td><td></td></tr> <tr><td><input type="checkbox"/></td><td>%Q0.2</td><td></td><td></td><td><input type="checkbox"/></td><td>0</td><td></td></tr> <tr><td><input type="checkbox"/></td><td>%Q0.3</td><td></td><td></td><td><input type="checkbox"/></td><td>0</td><td></td></tr> <tr><td><input type="checkbox"/></td><td>%Q0.4</td><td></td><td></td><td><input type="checkbox"/></td><td>0</td><td></td></tr> <tr><td><input type="checkbox"/></td><td>%Q0.5</td><td></td><td></td><td><input type="checkbox"/></td><td>0</td><td></td></tr> <tr><td><input type="checkbox"/></td><td>%Q0.6</td><td></td><td></td><td><input type="checkbox"/></td><td>0</td><td></td></tr> <tr><td><input type="checkbox"/></td><td>%Q0.7</td><td></td><td></td><td><input type="checkbox"/></td><td>0</td><td></td></tr> </tbody> </table> <p style="text-align: right;"> <input type="button" value="Apply"/> <input type="button" value="Cancel"/> </p> </div>	Used	Address	Symbol	Used by	Status Alarm	Fallback value	Comment	<input type="checkbox"/>	%Q0.0			<input type="checkbox"/>	0		<input type="checkbox"/>	%Q0.1			<input type="checkbox"/>	0		<input type="checkbox"/>	%Q0.2			<input type="checkbox"/>	0		<input type="checkbox"/>	%Q0.3			<input type="checkbox"/>	0		<input type="checkbox"/>	%Q0.4			<input type="checkbox"/>	0		<input type="checkbox"/>	%Q0.5			<input type="checkbox"/>	0		<input type="checkbox"/>	%Q0.6			<input type="checkbox"/>	0		<input type="checkbox"/>	%Q0.7			<input type="checkbox"/>	0	
Used	Address	Symbol	Used by	Status Alarm	Fallback value	Comment																																																										
<input type="checkbox"/>	%Q0.0			<input type="checkbox"/>	0																																																											
<input type="checkbox"/>	%Q0.1			<input type="checkbox"/>	0																																																											
<input type="checkbox"/>	%Q0.2			<input type="checkbox"/>	0																																																											
<input type="checkbox"/>	%Q0.3			<input type="checkbox"/>	0																																																											
<input type="checkbox"/>	%Q0.4			<input type="checkbox"/>	0																																																											
<input type="checkbox"/>	%Q0.5			<input type="checkbox"/>	0																																																											
<input type="checkbox"/>	%Q0.6			<input type="checkbox"/>	0																																																											
<input type="checkbox"/>	%Q0.7			<input type="checkbox"/>	0																																																											
2	<p>Edit the properties to configure the digital outputs. For detailed information on the digital output configuration parameters, refer to the table below.</p>																																																															

This table describes each parameter of the digital output configuration:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	True/False	False	Indicates whether the output channel is being used in a program or not.
<b>Address</b>	No	%Q0.x	–	Displays the address of the digital output on the controller, where x represents the channel number. If the controller has 8 digital output channels, x varies from 0...7. If the controller has 16 digital output channels, x varies from 0...15. For example, %Q0.2 is the third digital output channel on the controller.
<b>Symbol</b>	Yes	–	–	Allows you to specify a symbol to associate with the digital output object. Double-click in the <b>Symbol</b> column, type the name of the symbol and press <b>Enter</b> .
<b>Used by</b>	No	<i>any</i>	<i>empty</i>	Displays the name of the component that uses the output channel. For example, if the output channel is used as status alarm, it displays <b>Alarm</b> .
<b>Status Alarm</b>	Yes	True/False	False	Allows you to enable or disable the status alarm for the output (%Q0.0...%Q0.7). You can configure only one output channel for the status alarm. You cannot configure an output as status alarm if the output is used in a program. The value of the status alarm is 1 when the controller is in the state <code>RUNNING</code> , and 0 in all other state
<b>Fallback value</b>	Yes	1 or 0	0	Specify the value to apply to this output (fallback to 0 or fallback to 1) when the logic controller enters the <code>STOPPED</code> or an exception state. The default value is 0. If <b>Maintain values</b> fallback mode is configured, the output retains its current value when the logic controller enters the <code>STOPPED</code> or an exception state. This field is disabled for the output configured as <b>Status Alarm</b> .
<b>Comment</b>	Yes	–	–	Allows you to specify a comment to associate with the digital output object. Double-click in the <b>Comment</b> column, type the comment and press <b>Enter</b> .

Additional configuration details are displayed in the **Programming** tab. For more information, refer to Digital Outputs (%Q) ([see page 165](#)).

# Section 4.3

## Analog Input Configuration

### Configuring Analog Inputs

#### Introduction

The analog inputs do not have any configurable property in SoMachine Basic. By default, analog inputs are used as regular inputs.

#### Analog Inputs Configuration

This table describes how to configure the analog inputs:

Step	Action																																				
1	<p>Click the <b>Analog inputs</b> node in the hardware tree to display the analog input properties. This figure shows the properties of the analog inputs in the editor area:</p> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p><b>Analog inputs</b></p> <table border="1"> <thead> <tr> <th>Used</th> <th>Address</th> <th>Symbol</th> <th>Type</th> <th>Scope</th> <th>Minimum</th> <th>Maximum</th> <th>Filter</th> <th>Filter Unit</th> <th>Sampling</th> <th>Units</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>%IW0.0</td> <td></td> <td>0 - 10 V</td> <td>Normal</td> <td>0</td> <td>1000</td> <td>0</td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%IW0.1</td> <td></td> <td>0 - 10 V</td> <td>Normal</td> <td>0</td> <td>1000</td> <td>0</td> <td></td> <td></td> <td></td> <td></td> </tr> </tbody> </table> <p style="text-align: right;"> <input type="button" value="Apply"/> <input type="button" value="Cancel"/> </p> </div>	Used	Address	Symbol	Type	Scope	Minimum	Maximum	Filter	Filter Unit	Sampling	Units	Comment	<input type="checkbox"/>	%IW0.0		0 - 10 V	Normal	0	1000	0					<input type="checkbox"/>	%IW0.1		0 - 10 V	Normal	0	1000	0				
Used	Address	Symbol	Type	Scope	Minimum	Maximum	Filter	Filter Unit	Sampling	Units	Comment																										
<input type="checkbox"/>	%IW0.0		0 - 10 V	Normal	0	1000	0																														
<input type="checkbox"/>	%IW0.1		0 - 10 V	Normal	0	1000	0																														
2	<p>Edit the properties to configure the analog inputs. For detailed information on the analog input configuration parameters, refer to the table below.</p>																																				

This table describes each parameter of the analog input configuration:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	True/False	False	Indicates whether the input channel is being used in a program or not.
<b>Address</b>	No	%IW0.x	–	Displays the address of the analog input on the controller, where x represents the channel number. If the controller has 2 analog input channels, x is either 0 or 1. For example, %IW0.1 is the second analog input channel on the controller.
<b>Symbol</b>	Yes	–	–	Allows you to specify a symbol to associate with the analog input object. Double-click in the <b>Symbol</b> column, type the name of the symbol and press <b>Enter</b> .
<b>Type</b>	No	<b>0 - 10 V</b>	<b>0 - 10 V</b>	Indicates the channel mode. For example, <b>0 - 10 V</b> refers to the channel that can be used for an electrical input of voltage type in the range 0...10 V.
<b>Scope</b>	No	<b>Normal</b>	<b>Normal</b>	Indicates the range of values for a channel.
<b>Minimum</b>	No	0	0	Indicates the lower measurement limit.
<b>Maximum</b>	No	1000	1000	Indicates the upper measurement limit.
<b>Filter</b>	No	0	0	Indicates the filtering value. Multiply by the <b>Filter Unit</b> value to obtain the filtering time.
<b>Filter Unit</b>	No	100 ms	<i>empty</i>	Specifies the unit of time for the filtering value.
<b>Sampling</b>	No	–	<i>empty</i>	–
<b>Units</b>	No	<i>any</i>	<i>empty</i>	Indicates the unit of the analog input.
<b>Comment</b>	Yes	–	–	Allows you to specify a comment to associate with the analog input object. Double-click in the <b>Comment</b> column, type the comment and press <b>Enter</b> .

Additional configuration details are displayed in the **Programming** tab. For more information, refer to Analog Inputs (%IW) ([see page 166](#)).

## Section 4.4

### High Speed Counter Configuration

---

#### What Is in This Section?

This section contains the following topics:

Topic	Page
Configuring High Speed Counters	89
Configuring Dual Phase and Single Phase Counters	92
Configuring Frequency Meter	96

## Configuring High Speed Counters

### Introduction

You can configure high speed counters to perform any one of the following functions:

- Single Phase
- Dual Phase [Pulse / Direction]
- Dual Phase [Quadrature X1]
- Dual Phase [Quadrature X2]
- Dual Phase [Quadrature X4]
- Frequency Meter

For information on how to select a function, refer to High Speed Counter in Counter Modes or High Speed Counter in Frequency Meter Mode (see *Modicon M221 Logic Controller, Advanced Functions Library Guide*).

The **High Speed Counter** function block works at a maximum frequency of 100 kHz for all counting modes with a range of 0 to 65535 in single word and 0 to 4294967295 in double word.

The **High Speed Counter** function blocks use dedicated inputs and auxiliary inputs and outputs. These inputs and outputs are not reserved for the exclusive use of **High Speed Counter** function blocks:

- If the dedicated input/output is not used by an HSC instance, it is available for the application as a regular digital input/output.
- If the application does not use an HSC dedicated input/output as a regular digital input/output, it is available for the corresponding HSC instance.

### Single Phase I/O Assignment

	Main Inputs		Auxiliary Inputs		Reflex Outputs	
%HSC0	%I0.0	-	%I0.2	%I0.3	%Q0.2	%Q0.3
%HSC1	%I0.6	-	%I0.5	%I0.4	%Q0.4	%Q0.5
%HSC2	%I0.1	-	-	-	%Q0.2	%Q0.3
%HSC3	%I0.7	-	-	-	%Q0.4	%Q0.5
Single Phase	Pulse input	Not used	Preset input	Catch input	Reflex output 0	Reflex output 1

### Dual Phase Pulse / Direction I/O Assignment

	Main Inputs		Auxiliary Inputs		Reflex Outputs	
%HSC0	%I0.0	%I0.1	%I0.2	%I0.3	%Q0.2	%Q0.3
%HSC1	%I0.6	%I0.7	%I0.5	%I0.4	%Q0.4	%Q0.5
Pulse / Direction	Pulse input	Direction input	Preset input	Catch input	Reflex output 0	Reflex output 1

### Dual Phase Quadrature I/O Assignment

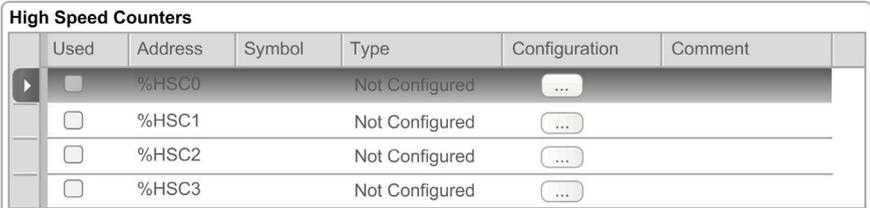
	Main Inputs		Auxiliary Inputs		Reflex Outputs	
%HSC0	%I0.0	%I0.1	%I0.2	%I0.3	%Q0.2	%Q0.3
%HSC1	%I0.6	%I0.7	%I0.5	%I0.4	%Q0.4	%Q0.5
Quadrature X1	Pulse input Phase A	Pulse input Phase B	Preset input	Catch input	Reflex output 0	Reflex output 1
Quadrature X2	Pulse input Phase A	Pulse input Phase B	Preset input	Catch input	Reflex output 0	Reflex output 1
Quadrature X4	Pulse input Phase A	Pulse input Phase B	Preset input	Catch input	Reflex output 0	Reflex output 1

### Frequency Meter I/O Assignment

	Main Inputs		Auxiliary Inputs		Reflex Outputs	
%HSC0	%I0.0	-	-	-	-	-
%HSC1	%I0.6	-	-	-	-	-
Frequency Meter	Pulse input	Not used	Not used	Not used	Not used	Not used

### High Speed Counters Configuration

This table describes how to configure the high speed counters:

Step	Description																														
1	<p>Click the <b>High Speed Counters</b> node in the hardware tree.  <b>Result:</b> The <b>High Speed Counters</b> list is displayed:</p>  <table border="1" data-bbox="216 980 1086 1188"> <thead> <tr> <th>Used</th> <th>Address</th> <th>Symbol</th> <th>Type</th> <th>Configuration</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/></td> <td>%HSC0</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%HSC1</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%HSC2</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%HSC3</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> </tbody> </table>	Used	Address	Symbol	Type	Configuration	Comment	<input checked="" type="checkbox"/>	%HSC0		Not Configured	...		<input type="checkbox"/>	%HSC1		Not Configured	...		<input type="checkbox"/>	%HSC2		Not Configured	...		<input type="checkbox"/>	%HSC3		Not Configured	...	
Used	Address	Symbol	Type	Configuration	Comment																										
<input checked="" type="checkbox"/>	%HSC0		Not Configured	...																											
<input type="checkbox"/>	%HSC1		Not Configured	...																											
<input type="checkbox"/>	%HSC2		Not Configured	...																											
<input type="checkbox"/>	%HSC3		Not Configured	...																											
2	<p>Click ... in under <b>Configuration</b> to select the type of high speed counter to assign and to display the <b>High Speed Counter Assistant</b> window.                      For detailed information on the high speed counter, refer to the table below.</p>																														

This table describes each parameter of the high speed counters configuration:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	TRUE/FALSE	FALSE	Indicates whether the high speed counter is being used in a program or not.
<b>Address</b>	No	%HSCi		Indicates the address of the high speed counter, where <i>i</i> is the object number.
<b>Symbol</b>	Yes	–	–	Allows you to specify a symbol to associate with the high speed counter object. Double-click in the <b>Symbol</b> column to edit the field.
<b>Type</b>	No	Not Configured Single Phase Dual Phase Frequency Meter	Not Configured	Indicates the counter operational mode.
<b>Configuration</b>	Yes	[...] (Button)	Disabled	Allows you to configure the high speed counter parameters using the <b>High Speed Counter Assistant</b> window.
<b>Comment</b>	Yes	–	–	Allows you to specify a comment to associate with the high speed counter object. Double-click in the <b>Comment</b> column to edit the field.

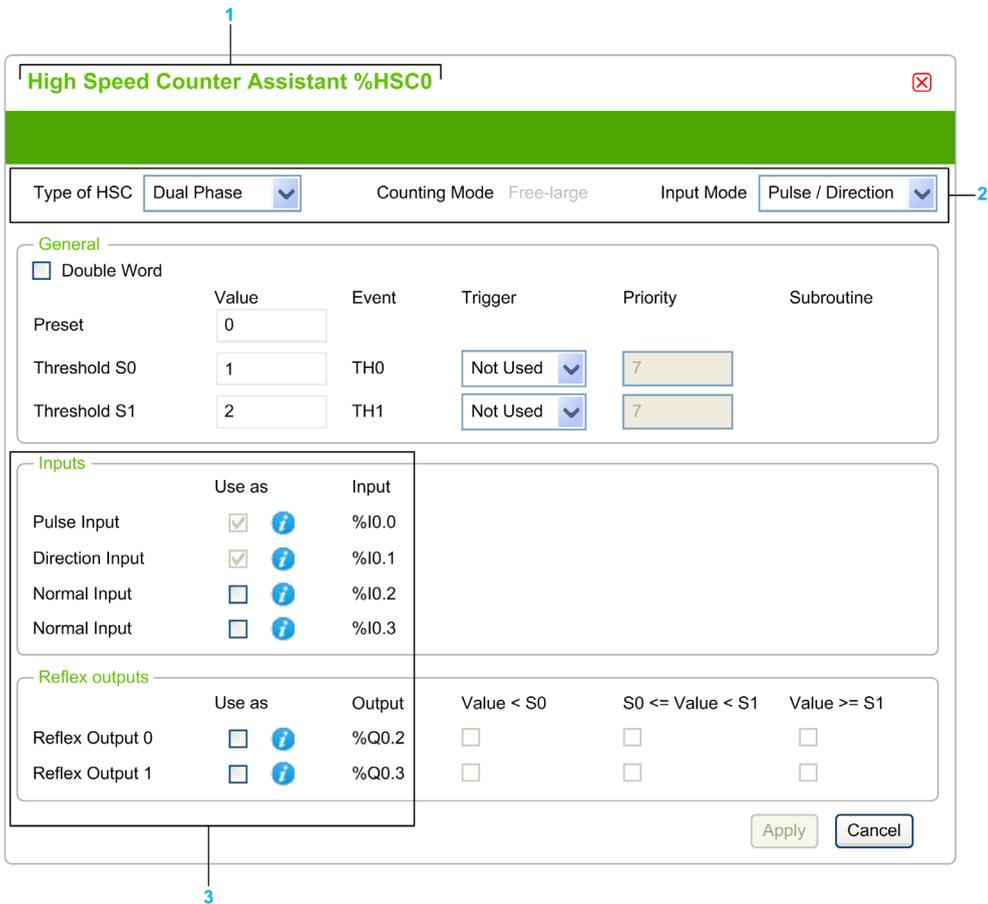
For details on the configuration of the Dual Phase [Pulse / Direction], Dual Phase [Quadrature X1], Dual Phase [Quadrature X2], Dual Phase [Quadrature X4], and Single Phase, refer to [Configuring Dual Phase and Single Phase Counters \(see page 92\)](#).

For details on the configuration of the Frequency Meter, refer to [Configuring Frequency Meter \(see page 96\)](#).

## Configuring Dual Phase and Single Phase Counters

### High Speed Counter Assistant

This figure presents an instance of the assistant window for %HSC0 configured as the Dual Phase [Pulse / Direction]:



Item	Description
1	Displays the title of the assistant dialog window for the selected HSC instance %HSCi.
2	Allows you to select the HSC type, mode, and dual phase counter type.
3	Displays the dedicated inputs, auxiliary inputs, and reflex outputs. Properties in this area of the assistant window are different for each counter type and the HSC instance. For more details, refer to Dedicated I/O Assignments ( <a href="#">see page 89</a> ).

### Common Parameters

This table describes parameters common to the counter types: Dual Phase [Pulse / Direction], Dual Phase [Quadrature X2], and Single Phase:

Parameter	Editable	Value	Default Value	Description
<b>Type of HSC</b>	Yes	Not Configured Single Phase Dual Phase Frequency Meter	-	Indicates the selected counter operational mode and allows you to change it. The options depend on the instance and on the type of HSC in the other instances. Refer to Dedicated I/O Assignments ( <a href="#">see page 89</a> ).
<b>Counting Mode</b>	No	Free Large	-	Indicates the selected counter operational mode. The options depend on the instance and on the type of HSC in the other instances. Refer to Dedicated I/O Assignments ( <a href="#">see page 89</a> ).
<b>Input Mode</b>	Yes	Pulse / Direction Quadrature X1 Quadrature X2 Quadrature X4	-	Indicates the selected counter operational mode and allows you to change it. The options depend on the instance and on the type of HSC in the other instances. Refer to Dedicated I/O Assignments ( <a href="#">see page 89</a> ).
<b>Double Word</b>	Yes	TRUE/FALSE	FALSE	Allows you to toggle between input data size of Word (16 bits) and Double Word (32 bits). Enabling this field changes the data size from Word (16 bits) to Double Word (32 bits).
<b>Preset</b>	Yes	0...65535 (Word)	0 (Word)	Allows you to specify the preset value for counting functions.
		0...4294967295 (Double Word)	0 (Double Word)	
<b>Threshold S0</b>	Yes	0...65535 (Word)	65535 (Word)	Allows you to specify the value of the HSC flag S0 that contains the value of the threshold TH0.
		0...4294967295 (Double Word)	4294967295 (Double Word)	
<b>Threshold S1</b>	Yes	0...65535 (Word)	0...65535 (Word)	Allows you to specify the value for the HSC flag S1 that contains the value of the threshold TH1.
		0...4294967295 (Double Word)	0... 4294967295 (Double Word)	

Parameter	Editable	Value	Default Value	Description
<b>Trigger</b>	Yes	<b>Not Used</b> <b>Falling Edge</b> <b>Rising Edge</b> <b>Both edges</b>	<b>Not Used</b>	Allows you to select a triggering function for an event (for both threshold TH0 and TH1) from the list. Selecting a triggering function makes the <b>Priority</b> parameter editable.
<b>Priority</b>	Yes	0...7	7	Allows you to set the priority of the triggering function of an event (for both threshold TH0 and TH1). This field is greyed until you select a triggering function.
<b>Subroutine</b>	No	<i>any</i>	<i>empty</i>	Displays the subroutine associated with an input configured as an event (for both threshold TH0 and TH1).
<b>Normal Input</b>	Yes	TRUE/FALSE	FALSE	Configurable as <b>Preset Input</b> by selecting the <b>Use as</b> check box, only on %HSC0 and %HSC1, respectively %I0.2 and %I0.5.
<b>Normal Input</b>	Yes	TRUE/FALSE	FALSE	Configurable as <b>Catch Input</b> by selecting the <b>Use as</b> check box, only on %HSC0 and %HSC1, respectively %I0.3 and %I0.4.
<b>Reflex Output 0</b>	Yes	TRUE/FALSE	FALSE	Configure Reflex output 0 %Q0.2 for either %HSC0 or %HSC2. Configure Reflex output 0 %Q0.4 for either %HSC1 or %HSC3.
<b>Reflex Output 1</b>	Yes	TRUE/FALSE	FALSE	Configure Reflex output 1 %Q0.3 for either %HSC0 or %HSC2. Configure Reflex output 1 %Q0.5 for either %HSC1 or %HSC3.
<b>Value &lt; S0</b>	Yes	TRUE/FALSE	FALSE	Allows you to enable or disable the condition in which the counter is constantly compared to the output value to set the reflex output when the output value is lower than the value of HSC flag S0.
<b>S0 &lt;= Value &lt; S1</b>	Yes	TRUE/FALSE	FALSE	Allows you to enable or disable the condition in which the counter is constantly compared to the output value to set the reflex output when the output value is higher than or equals to the value of the HSC flag S0 and the output value is lower than the value of the HSC flag S1.
<b>Value &gt;= S1</b>	Yes	TRUE/FALSE	FALSE	Allows you to enable or disable the condition in which the counter is constantly compared to the output value to set the reflex output when the output value is higher than or equals to the value of HSC flag S1.

### Dual Phase [Pulse / Direction] Parameters

This table describes parameters specific to Dual Phase [Pulse / Direction]:

Parameter	Editable	Value	Default Value	Description
Pulse Input	No	TRUE/FALSE	TRUE	Configured as pulse input, only on %HSC0 and %HSC1, respectively %I0.0 and %I0.6.
Direction Input	No	TRUE/FALSE	TRUE	Configured as directional input, only on %HSC0 and %HSC1, respectively %I0.1 and %I0.7. <ul style="list-style-type: none"> <li>● TRUE = down counting</li> <li>● FALSE = up counting</li> </ul>

### Dual Phase [Quadrature X1], Dual Phase [Quadrature X2], and Dual Phase [Quadrature X4] Parameters

This table describes parameters specific to Dual Phase [Quadrature X1], Dual Phase [Quadrature X2], and Dual Phase [Quadrature X4]:

Parameter	Editable	Value	Default Value	Description
Pulse Input Phase A	No	TRUE/FALSE	TRUE	Configured as pulse input for phase A, only on %HSC0 and %HSC1, respectively %I0.0 and %I0.6.
Pulse Input Phase B	No	TRUE/FALSE	TRUE	Configured as pulse input for phase B, only on %HSC0 and %HSC1, respectively %I0.1 and %I0.7.

### Single Phase Parameters

This table describes a parameter specific to Single Phase:

Parameter	Editable	Value	Default Value	Description
Pulse Input	No	TRUE/FALSE	TRUE	You can configure up to four HSC in Single Phase HSC type, they use, as a pulse input: <ul style="list-style-type: none"> <li>● %I0.0 for %HSC0</li> <li>● %I0.6 for %HSC1</li> <li>● %I0.1 for %HSC2</li> <li>● %I0.7 for %HSC3</li> </ul>

## Configuring Frequency Meter

### High Speed Counter Assistant

This figure presents the **High Speed Counter Assistant (%HSC0)** window for the counter type Frequency Meter:

**High Speed Counter Assistant %HSC0**

Type of HSC: Frequency Meter

**General**

Double Word

Time Window

100 ms

1 s

**Inputs**

Pulse Input	Use as	Input
Pulse Input	<input checked="" type="checkbox"/>	%I0.0

Apply Cancel

### Frequency Meter Parameters

This table describes each parameter of the **High Speed Counter Assistant (%HSCi)** window for the counter type `Frequency Meter`:

Parameter	Editable	Value	Default Value	Description
<b>Type of HSC</b>	Yes	Not Configured Single Phase Dual Phase Frequency Meter	Frequency Meter	Indicates the selected counter operational mode and allows you to change it. The Frequency Meter is configurable on %HSC0 and/or %HSC1. Refer to the Frequency Meter I/O Assignment ( <i>see page 90</i> ).
<b>Double Word</b>	Yes	TRUE/FALSE	FALSE	Use a 32-bit preset word. Enabling this field changes the data size from Word (16 bits) to Double Word (32 bits).
<b>Time Window</b>	Yes	100 ms 1 s	1 s	Allows you to select the time base to measure the frequency between 100 Hz and 100 kHz.
<b>Pulse Input</b>	No	TRUE/FALSE	TRUE	Indicates the input used as pulse input, %I0.0 for %HSC0 or %I0.6 for %HSC1.

Additional configuration details are displayed in the **Programming** tab.

For more details on the `High Speed Counter` function block, refer to the Modicon M221 Logic Controller Advanced Functions Library Guide, chapter High Speed Counter Function Block (%HSC).

## Section 4.5

### Pulse Generator Configuration

---

#### What Is in This Section?

This section contains the following topics:

Topic	Page
Configuring Pulse Generators	99
Configuring Pulse (%PLS)	102
Configuring Pulse Width Modulation (%PWM)	104
Configuring Pulse Train Output (%PTO)	106
Configuring Frequency Generator (%FREQGEN)	109

## Configuring Pulse Generators

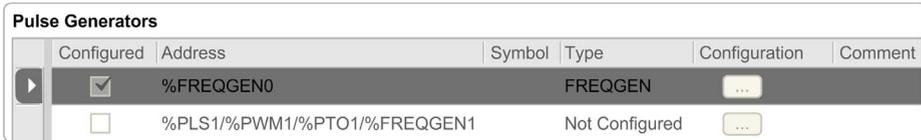
### Introduction

The pulse generator function blocks, Pulse (PLS), Pulse Width Modulation (PWM), Pulse Train Output (PTO) and Frequency Generator (FREQGEN) are used to generate square or modulated wave signals on dedicated output channels %Q0.0 or %Q0.1.

The PWM outputs feature a modulated wave signal with a variable width and duty cycle, while the PTO outputs generate a square wave to control a linear single-axis stepper or servo drive in open loop mode. The PLS also creates a square wave for a programmed number of pulses.

### Pulse Generators Configuration

This table describes how to configure the pulse generators:

Step	Action																		
1	<p>Click the <b>Pulse Generators</b> node in the hardware tree to display the pulse generator properties. This figure presents the properties of the pulse generators in the editor area:</p>  <table border="1" data-bbox="319 678 1240 818"> <thead> <tr> <th>Configured</th> <th>Address</th> <th>Symbol</th> <th>Type</th> <th>Configuration</th> <th>Comment</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/></td> <td>%FREQGEN0</td> <td></td> <td>FREQGEN</td> <td>...</td> <td></td> </tr> <tr> <td><input type="checkbox"/></td> <td>%PLS1/%PWM1/%PTO1/%FREQGEN1</td> <td></td> <td>Not Configured</td> <td>...</td> <td></td> </tr> </tbody> </table>	Configured	Address	Symbol	Type	Configuration	Comment	<input checked="" type="checkbox"/>	%FREQGEN0		FREQGEN	...		<input type="checkbox"/>	%PLS1/%PWM1/%PTO1/%FREQGEN1		Not Configured	...	
Configured	Address	Symbol	Type	Configuration	Comment														
<input checked="" type="checkbox"/>	%FREQGEN0		FREQGEN	...															
<input type="checkbox"/>	%PLS1/%PWM1/%PTO1/%FREQGEN1		Not Configured	...															
2	<p>Edit the properties and click [...] to configure the pulse generator output. For detailed information on the pulse generator configuration parameters, refer to the table below.</p>																		

This table describes the parameters of the pulse generator:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	True/False	False	Indicates whether the pulse generated output is being used in a program or not.
<b>Address</b>	No	%PLSx %PWMx %PTOx %FREQGENx	%PLSx/%PWMx/ %PTOx/%FREQGENx	Displays the address of the Pulse output, Pulse Width Modulation output, Pulse Train Output, or Frequency Generator where x is the output number.
<b>Symbol</b>	Yes	–	–	Allows you to specify a symbol to associate with the pulse generator object. Double-click in the <b>Symbol</b> column to edit the field.
<b>Type</b>	No	<b>Not Configured</b> <b>PLS</b> <b>PWM</b> <b>PTO</b> <b>FREQGEN</b>	<b>Not Configured</b>	Displays the type of the pulse generator used for the output channel.
<b>Configuration</b>	Yes	[...] (Button)	Enabled	Allows you to configure the pulse generator using the <b>Pulse Generator Assistant</b> window.
<b>Comment</b>	Yes	–	–	Allows you to specify a comment to associate with the pulse generator object. Double-click in the <b>Comment</b> column to edit the field.

### PLS Configuration

Refer to Configuring Pulse (%PLS) ([see page 102](#)).

For more details on the Pulse function block, refer to the Modicon M221 Logic Controller Advanced Functions Library Guide, chapter Pulse (%PLS).

### PWM Configuration

Refer to Configuring Pulse Width Modulation (%PWM) ([see page 104](#)).

For more details on the Pulse Width Modulation function block, refer to the Modicon M221 Logic Controller Advanced Functions Library Guide, chapter Pulse Width Modulation (%PWM).

### PTO Configuration

Refer to Configuring Pulse Train Output (%PTO) ([see page 106](#)).

For more details on the Pulse Train Output function block, refer to the Modicon M221 Logic Controller Advanced Functions Library Guide, chapter Pulse Train Output (%PTO).

## Frequency Generator Configuration

Refer to Configuring Frequency Generator (%FREQGEN) (*see page 106*)

For more details on the `FREQGEN` function block, refer to the Modicon M221 Logic Controller Advanced Functions Library Guide, chapter Frequency Generator (%FREQGEN).

## Configuring Pulse (%PLS)

### Pulse Generator Assistant for PLS

This graphic presents the **Pulse Generator Assistant** window when the **Type of pulse generator** is set to **PLS**:

**Pulse Generator Assistant %PLS0** [Close]

**General** Type of pulse generator **PLS**  %Q0.0

**Behavior**  Double Word

**Period** Time Base **1 s**  
Preset **0**

[Apply] [Cancel]

The table describes each parameter available when the channel is configured in **PLS** mode:

Parameter	Value	Default Value	Description
<b>Type of pulse generator</b>	Not Configured PLS PWM PTO FREQGEN	PLS	Allows you to choose the type of pulse generator and configure the output properties. Select: <ul style="list-style-type: none"> <li>● <b>PLS</b> to configure the output channels in <i>PLS</i> mode. Refer to Configuring Pulse (%PLS) (<a href="#">see page 102</a>).</li> <li>● <b>PWM</b> to configure the output channels in <i>PWM</i> mode. Refer to Configuring Pulse Width Modulation (%PWM) (<a href="#">see page 104</a>).</li> <li>● <b>PTO</b> to configure the output channels in <i>PTO</i> mode. Refer to Configuring Pulse Train Output (%PTO) (<a href="#">see page 106</a>).</li> <li>● <b>FREQGEN</b> to configure the output channels in <i>FREQGEN</i> mode. Refer to Configuring Frequency Generator (%FREQGEN) (<a href="#">see page 109</a>).</li> </ul>
<b>Double Word</b>	False	True/False	Allows you to toggle between the data size of Word (16 bits) and Double Word (32 bits). By default, this parameter is disabled, which indicates that the current data size is Word (16 bits). Enabling this field changes the data size to Double Word (32 bits).
<b>Time Base</b>	0.1 ms 1 ms 10 ms 1 s	1 s	Allows you to select the time base for the frequency measurement.
<b>Preset</b>	Refer to the table below for complete range of preset values for <i>PLS</i> type pulse generator.	0	Allows you to specify the preset value for the pulse output.

This table presents the range of values of the **Preset** parameter:

Type	Time Base	Preset Value Range
PLS	0.1 ms	1...20000
	1 ms	1...2000
	10 ms	1...200
	1 s	1 or 2

Additional configuration details are displayed in the **Programming** tab.

For more details on the `Pulse` function block, refer to the Modicon M221 Logic Controller Advanced Functions Library Guide, chapter Pulse (%PLS).

## Configuring Pulse Width Modulation (%PWM)

### Pulse Generator Assistant for PWM

This graphic presents the **Pulse Generator Assistant** window when the **Type of pulse generator** is set to **PWM**:

**Pulse Generator Assistant %PWM0** ✕

---

**General** Type of pulse generator PWM ▾  %Q0.0

**Period** Time Base 1 s ▾  
Preset

Apply Cancel

The table describes each parameter available when the channel is configured in **PWM** mode:

Parameter	Value	Default Value	Description
<b>Type of pulse generator</b>	<b>Not Configured</b> <b>PLS</b> <b>PWM</b> <b>PTO</b> <b>FREQGEN</b>	<b>PWM</b>	Allows you to choose the type of pulse generator and configure the output properties. Select: <ul style="list-style-type: none"> <li>● <b>PLS</b> to configure the output channels in <b>PLS</b> mode. Refer to Configuring Pulse (%PLS) (<a href="#">see page 102</a>).</li> <li>● <b>PWM</b> to configure the output channels in <b>PWM</b> mode. Refer to Configuring Pulse Width Modulation (%PWM) (<a href="#">see page 104</a>).</li> <li>● <b>PTO</b> to configure the output channels in <b>PTO</b> mode. Refer to Configuring Pulse Train Output (%PTO) (<a href="#">see page 106</a>).</li> <li>● <b>FREQGEN</b> to configure the output channels in <b>FREQGEN</b> mode. Refer to Configuring Frequency Generator (%FREQGEN) (<a href="#">see page 109</a>).</li> </ul>
<b>Time Base</b>	<b>0.1 ms</b> <b>1 ms</b> <b>10 ms</b> <b>1 s</b>	<b>1 s</b>	Allows you to select the time base for the frequency measurement.
<b>Preset</b>	Refer to the table below for complete range of preset values for <b>PWM</b> type pulse generator.	0	Allows you to specify the preset value for the <b>PWM</b> output.

This table presents the range of values of the **Preset** parameter:

Type	Time Base	Preset Value Range
PWM	0.1 ms	1...10000
	1 ms	1...1000
	10 ms	1...100
	1 s	1

Additional configuration details are displayed in the **Programming** tab.

For more details on the **Pulse Width Modulation** function block, refer to the Modicon M221 Logic Controller Advanced Functions Library Guide, chapter Pulse Width Modulation (%PWM).

## Configuring Pulse Train Output (%PTO)

### Pulse Generator Assistant for PTO

This graphic presents the **Pulse Generator Assistant** window when the **Type of pulse generator** is set to **PTO**:

**Pulse Generator Assistant %PTO0**
✕

---

General

Type of pulse generator: PTO

Pulse: %Q0.0

Output mode: Pulse / Direction

Direction: %Q0.4

---

Mechanics

Backlash Compensation: 0

---

Software Position Limits

Enable the software position limits  

Zone of operation

Low limit: -2147483648

High limit: 2147483647

---

Motion

Max. velocity (Hz): 100000  
 Start velocity (Hz): 0  
 Stop velocity (Hz): 0

Max. acc. (Hz/ms): 100000

Fast stop dec. (Hz/ms): 5000

Max dec. (Hz/ms): 100000

---

Homing

REF input: Not Used  
 Contact type: Normally opened

---

Probe activation

PROBE input: Not Used

Apply Cancel

The table describes each parameter available when the channel is configured in **PTO** mode:

Parameter		Value	Default	Description
General	<b>Type of pulse generator</b>	<b>Not Configured</b> <b>PLS</b> <b>PWM</b> <b>PTO</b> <b>FREQGEN</b>	<b>PTO</b>	Allows you to choose the type of pulse generator and configure the output properties. Select: <ul style="list-style-type: none"> <li>● <b>PLS</b> to configure the output channels in <b>PLS</b> mode. Refer to Configuring Pulse (%PLS) (<a href="#">see page 102</a>).</li> <li>● <b>PWM</b> to configure the output channels in <b>PWM</b> mode. Refer to Configuring Pulse Width Modulation (%PWM) (<a href="#">see page 104</a>).</li> <li>● <b>PTO</b> to configure the output channels in <b>PTO</b> mode. Refer to Configuring Pulse Train Output (%PTO) (<a href="#">see page 106</a>).</li> <li>● <b>FREQGEN</b> to configure the output channels in <b>FREQGEN</b> mode. Refer to Configuring Frequency Generator (%FREQGEN) (<a href="#">see page 109</a>).</li> </ul>
	Output mode	<b>Clock Wise / Counter</b> <b>Clock Wise</b> <b>Pulse / Direction</b>	<b>Pulse / Direction</b>	Select the pulse output mode. <b>NOTE:</b> The <b>Clock Wise / Counter Clock Wise</b> output mode is only valid for PTO0. This mode disables PTO1.
<b>Mechanics</b>	Backlash Compensation	0...65535	0	Set backlash compensation value. The specified number of backlash compensation pulses are not added to the position counter. See Backlash Compensation.
<b>Software Position Limits</b>	<b>Enable the software position limits</b>	Enabled Disabled	Enabled	Select whether to use the software position limits.
	<b>Low Limit</b>	<b>-2147483648...</b> <b>2147483647</b>	<b>-2147483648</b>	Set the software limit position to be detected in the negative direction.
	<b>High Limit</b>	<b>-2147483648...</b> <b>2147483647</b>	<b>2147483647</b>	Set the software limit position to be detected in the positive direction.

Parameter		Value	Default	Description
<b>Motion</b>	Max. velocity	0...100000	100000	Set the pulse output maximum velocity (in Hz).
	Start velocity	0...100000	0	Set the pulse output start velocity (in Hz). 0 if not used.
	Stop velocity	0...100000	0	Set the pulse output stop velocity (in Hz). 0 if not used.
	Max. acc.	1...100000	100000	Set the acceleration maximum value (in Hz/ms).
	Fast stop dec.	1...100000	5000	Set the deceleration value in case an error is detected (in Hz/ms)
	Max. dec.	1...100000	100000	Set the deceleration maximum value (in Hz/ms).
<b>Homing</b>	<b>REF input</b>	<b>Not Used</b> Input	<b>Not Used</b>	Select whether to use the REF input to set the homing position.
	<b>Contact type</b>	<b>Normally opened</b> <b>Normally closed</b>	<b>Normally opened</b>	Select whether the switch contact default state is open or closed.  <b>NOTE:</b> The input type is only available when <b>REF input</b> is selected.
<b>Probe activation</b>	<b>PROBE input</b>	<b>Not Used</b> Input	<b>Not Used</b>	Select whether to use the PROBE input.

Additional configuration details are displayed in the **Programming** tab.

For more details on the `Pulse Train Output` function block, refer to the Modicon M221 Logic Controller Advanced Functions Library Guide, chapter Pulse Train Output (%PTO).

## Configuring Frequency Generator (%FREQGEN)

### Pulse Generator Assistant for FREQGEN

This graphic presents the **Pulse Generator Assistant** window when the **Type of pulse generator** is set to **FREQGEN**:

**Pulse Generator Assistant %FREQGEN0** ✕

**General** Type of pulse generator **FREQGEN**  %Q0.0

**Frequency** Frequency (Hz)

The Frequency Generator (FG) function generates a square wave signal with programmable frequency and duty cycle of 50%. The controller uses an internal clock generator and provides an output signal on a dedicated output channel (%Q0.0). This output signal can directly command a constant motion of the axis. The target frequency is always positive.

For more details on the `FREQGEN` function block, refer to the Modicon M221 Logic Controller Advanced Functions Library Guide, chapter Frequency Generator (%FREQGEN).



---

# Chapter 5

## I/O Bus Configuration

---

### Overview

This chapter describes how to configure the I/O bus (expansion modules) of the M221 Logic Controller.

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
I/O Configuration General Practices	112
Maximum Hardware Configuration	113
Configuring Cartridges and Expansion Modules	117

## I/O Configuration General Practices

### Match Software and Hardware Configuration

The I/O that may be embedded in your controller is independent of the I/O that you may have added in the form of I/O expansion. It is crucial that the logical I/O configuration within your program matches the physical I/O configuration of your installation. If you add or remove any physical I/O to or from the I/O expansion bus, or, depending on the controller reference, to or from the controller (in the form of cartridges), it is imperative that you update your application configuration. This is also true for any field bus devices you may have in your installation. Otherwise, there is the possibility that the I/O expansions will no longer function while the embedded I/O that may be present in your controller will continue to operate.

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Update the configuration of your program each time you add or delete any type of I/O expansions on your I/O bus, or you add or delete any devices on your field bus.

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

### Presentation of the Optional Feature for I/O Expansion Modules

I/O expansion modules can be marked as optional in the configuration. The **Optional module** feature provides a more flexible configuration by the acceptance of the definition of modules that are not physically attached to the logic controller. Therefore, a single application can support multiple physical configurations of I/O expansion modules, allowing a greater degree of scalability without the necessity of maintaining multiple application files for the same application.

You must be fully aware of the implications and impacts of marking I/O modules as optional in your application, both when those modules are physically absent and present when running your machine or process. Be sure to include this feature in your risk analysis.

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

Include in your risk analysis each of the variations of I/O configurations that can be realized marking I/O expansion modules as optional, and in particular the establishment of TM3 Safety modules (TM3S...) as optional I/O modules, and make a determination whether it is acceptable as it relates to your application.

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

**NOTE:** For more details about this feature, refer to Optional I/O Expansion Modules ([see page 73](#)).

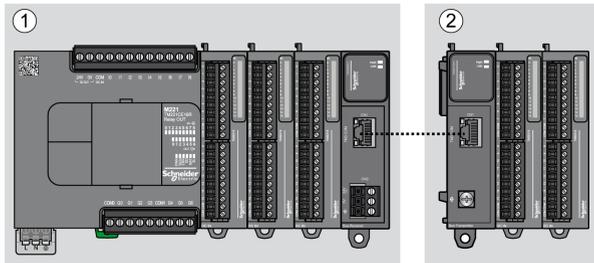
## Maximum Hardware Configuration

### Introduction

The M221 Logic Controller is a control system that offers an all-in-one solution with optimized configurations and an expandable architecture.

### Local and Remote Configuration Principle

The following figure defines the local and remote configurations:



- (1) Local configuration
- (2) Remote configuration

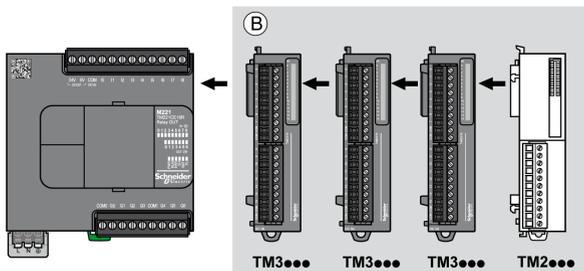
### M221 Logic Controller Local Configuration Architecture

Optimized local configuration and flexibility are provided by the association of:

- M221 Logic Controller
- TM3 expansion modules
- TM2 expansion modules

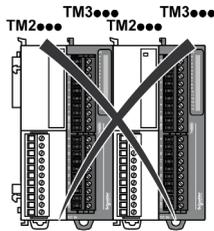
Application requirements determine the architecture of your M221 Logic Controller configuration.

The following figure represents the components of a local configuration:



- (B) Expansion modules (see maximum number of modules)

**NOTE:** You cannot mount a TM2 module before any TM3 module as indicated in the following figure:



### M221 Logic Controller Remote Configuration Architecture

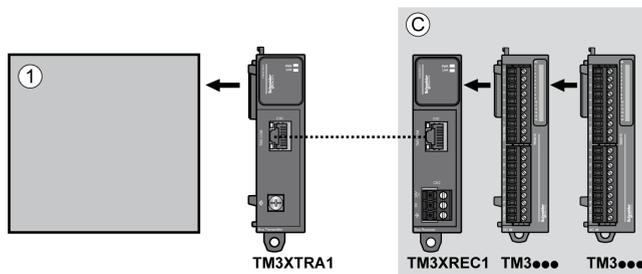
Optimized remote configuration and flexibility are provided by the association of:

- M221 Logic Controller
- TM3 expansion modules
- TM3 transmitter and receiver modules

Application requirements determine the architecture of your M221 Logic Controller configuration.

**NOTE:** You cannot use TM2 modules in configurations that include the TM3 transmitter and receiver modules.

The following figure represents the components of a remote configuration:



- (1) Logic controller and modules
- (C) Expansion modules (7 maximum)

## Maximum Number of Modules

The following table shows the maximum configuration supported:

References	Maximum	Type of Configuration
TM221C16• TM221CE16•	4 TM3 / TM2 expansion modules	Local
TM221C24• TM221CE24• TM221C40• TM221CE40• TM221M16R• TM221ME16R• TM221M16T• TM221ME16T• TM221M32TK TM221ME32TK	7 TM3 / TM2 expansion modules	Local
TM3XREC1	7 TM3 expansion modules	Remote
<b>NOTE:</b> TM3 transmitter and receiver modules are not included in a count of the maximum number of expansion modules.		

**NOTE:** The configuration with its TM3 and TM2 expansion modules is validated by SoMachine Basic software in the **Configuration** window taking into account the total power consumption of the installed modules.

**NOTE:** In some environments, the maximum configuration populated by high consumption modules, coupled with the maximum distance allowable between the TM3 transmitter and receiver modules, may present bus communication issues although the SoMachine Basic software allows for the configuration. In such a case you will need to analyze the consumption of the modules chosen for your configuration, as well as the minimum cable distance required by your application, and possibly seek to optimize your choices.

### Current Supplied to the I/O Bus

The following table shows the maximum current supplied by the controllers to the I/O Bus:

Reference	IO Bus 5 Vdc	IO Bus 24 Vdc
TM221C16R TM221CE16R	325 mA	120 mA
TM221C16T TM221CE16T	325 mA	148 mA
TM221C16U TM221CE16U	325 mA	148 mA
TM221C24R TM221CE24R	520 mA	160 mA
TM221C24T TM221CE24T	520 mA	200 mA
TM221C24U TM221CE24U	520 mA	200 mA
TM221C40R TM221CE40R	520 mA	240 mA
TM221C40T TM221CE40T	520 mA	304 mA
TM221C40U TM221CE40U	520 mA	304 mA
TM221M16R• TM221ME16R•	520 mA	460 mA
TM221M16T• TM221ME16T•	520 mA	492 mA
TM221M32TK TM221ME32TK	520 mA	484 mA

**NOTE:** Expansion modules consume current from the 5 Vdc and 24 Vdc supplied to the I/O Bus. Therefore, the current delivered by the logic controller to the I/O Bus defines the maximum number of expansion modules that can be connected to the I/O Bus (validated by SoMachine Basic software in the **Configuration** window).

## Configuring Cartridges and Expansion Modules

### Introduction

In your project, you can add the following devices to the controller:

- TMC2 Cartridges
- TM3 Digital I/O Modules
- TM3 Expert I/O Modules
- TM2 Digital I/O Modules
- TM2 Analog I/O Modules

### 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

### 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	



---

# Chapter 6

## Embedded Communication Configuration

---

### Overview

This chapter describes how to configure the communication features of the M221 Logic Controller.

### What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
6.1	Ethernet Configuration	120
6.2	Serial Line Configuration	133
6.3	Supported Modbus Function Codes	141

# Section 6.1

## Ethernet Configuration

---

### What Is in This Section?

This section contains the following topics:

Topic	Page
Configuring Ethernet Network	121
Configuring Modbus TCP	126
Configuring EtherNet/IP	130

## Configuring Ethernet Network

### Introduction

You can configure the TCP/IP connection to the logic controller by configuring the Ethernet network. The Ethernet establishes a local area network (LAN) between the logic controller and other devices. The Ethernet configuration provides you the ability to configure the IP address of the network device.

**NOTE:** The controller-PC link uses the TCP/IP protocol. It is required for this protocol to be installed on the PC.

You can obtain the IP address by the following protocols:

- Dynamic Host Configuration Protocol (DHCP)
- Bootstrap Protocol (BOOTP)

You can also specify the IP address by specifying the following addresses:

- IP address
- Subnet mask
- Gateway address

### Ethernet Services

The logic controller supports the following services:

- Modbus TCP Server
- Modbus TCP Client
- EtherNet/IP Adapter
- Modbus TCP Slave Device

This table presents the maximum number of TCP server connections:

Connection Type	Maximum Number of Connections
Modbus Server	8
Modbus Client	1
EtherNet/IP Device	8

Each server based on TCP manages its own set of connections.

When a client tries to open a connection that exceeds the poll size, the logic controller closes the oldest connection, other than the connection with SoMachine Basic.

The server connections stay open as long as the logic controller stays in its present operational state (RUNNING, STOPPED, or HALTED).

The server connections are closed when a transition is made from its present operational state (RUNNING, STOPPED, or HALTED), except in case of power outage (because the controller does not have time to close the connections).

The server connections can be closed when the EtherNet/IP originator or Modbus TCP Master requests to close.

## Ethernet Configuration

This table describes how to configure the Ethernet:

Step	Action
1	<p>Click the <b>ETH1</b> node in the hardware tree to display the Ethernet properties. This figure presents the Ethernet properties in the editor area:</p> <div style="border: 1px solid #ccc; padding: 10px;"> <p><b>Ethernet</b></p> <p>Device name <input type="text" value="M221"/></p> <p> <input type="radio"/> IP address by DHCP  <input type="radio"/> IP address by BOOTP  <input checked="" type="radio"/> Fixed IP address                 </p> <p>IP address <input type="text" value="0"/> . <input type="text" value="0"/> . <input type="text" value="0"/> . <input type="text" value="0"/></p> <p>Subnet mask <input type="text" value="0"/> . <input type="text" value="0"/> . <input type="text" value="0"/> . <input type="text" value="0"/></p> <p>Gateway address <input type="text" value="0"/> . <input type="text" value="0"/> . <input type="text" value="0"/> . <input type="text" value="0"/></p> <p>Transfer Rate <input type="button" value="Auto"/></p> <p><b>Security Parameters</b></p> <p><input checked="" type="checkbox"/> Programming protocol enabled</p> <p><input checked="" type="checkbox"/> EtherNet/IP protocol enabled</p> <p><input checked="" type="checkbox"/> Modbus server enabled</p> <p><input checked="" type="checkbox"/> Auto discovery protocol enabled</p> </div>
2	<p>Edit the properties to configure the Ethernet. For detailed information on the Ethernet configuration parameters, refer to the table below.</p>

**NOTE:** The **Security Parameters** displayed depend on the functional level selected for the application.

This table describes each parameter of the Ethernet configuration:

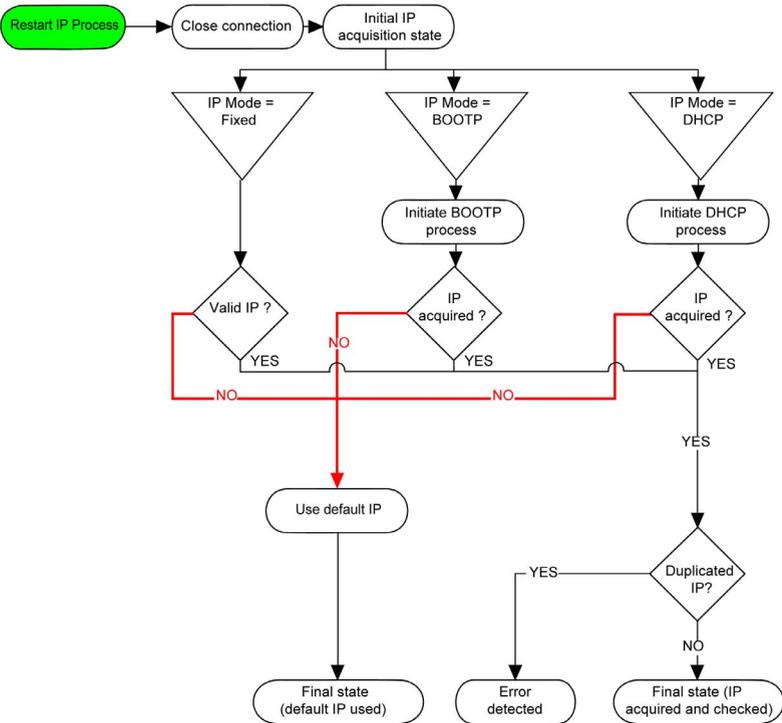
Parameter	Editable	Value	Default Value	Description
<b>Ethernet</b>				
<b>Device name</b>	No	<i>any</i>	<b>M221</b> (if the controller used in the configuration is M221 Logic Controller)	Displays the name of the device that is connected with the Ethernet network. The characters a...z, A...Z, 0...9 and the underscore character ( _ ) are allowed.
<b>IP address by DHCP</b>	Yes <sup>(1)</sup>	TRUE/FALSE	FALSE	Allows you to obtain the IP address from the DHCP server on the network.
<b>IP address by BOOTP</b>	Yes <sup>(1)</sup>	TRUE/FALSE	FALSE	Allows you to obtain the IP address from the Boot PROM configuration server on the network.
<b>Fixed IP address</b>	Yes <sup>(1)</sup>	TRUE/FALSE	TRUE	Allows you to specify the IP address manually for host or network interface identification.
<b>IP address</b>	Yes <sup>(2)</sup>	<i>w.x.y.z</i> <sup>(3)</sup>	0.0.0.0	Allows you to specify the IP address of the device in the Ethernet network. Assigning 0.0.0.0 as IP address for the M221 Logic Controller forces the firmware to generate an IP address from the MAC address. The generated IP address is 10.10.XXX.YYY, where XXX and YYY are the decimal values of the last 2 bytes (EE.FF) of the MAC address (AA.BB.CC.DD.EE.FF) Example: MAC address: 00:80:78:19:19:73 EE (19 hex) = <b>25</b> decimal FF (73 hex) = <b>155</b> decimal IP address generated: 10.10. <b>25.155</b> .
<b>Subnet mask</b>	Yes <sup>(2)</sup>	<i>w.x.y.z</i> <sup>(3)</sup>	0.0.0.0	Allows you to specify the address of the subnetwork to authorize a group of devices for data exchange. It determines which bits in an IP address correspond to the network address and which bits correspond to the subnet portions of the address.
<p><b>(1)</b> You can select any one option for IP addressing. Selecting any one option disables the other options.</p> <p><b>(2)</b> These options are enabled only if you select the option <b>Fixed IP Address</b> for IP addressing.</p> <p><b>(3)</b> w, x, y, and z are the bytes that store the address and each byte can store a value in the range 0...255.</p>				

Parameter	Editable	Value	Default Value	Description
<b>Gateway address</b>	Yes <sup>(2)</sup>	w.x.y.z <sup>(3)</sup>	0.0.0.0	Allows you to specify the IP address of the node (a router) on a TCP/IP network that serves as an access point to another network.
<b>Transfer Rate</b>	No	–	<b>Auto</b>	Displays the selected mode for Ethernet speed. Auto stands for “Auto Negotiation”.
<b>Security Parameters</b>				
The security parameters allow you to enable or disable communication protocols and features.				
<b>Programming protocol enabled</b>	Yes	TRUE/FALSE	TRUE	Allows you to enable or disable programming via the Ethernet port. Also enables or disables the software object access via animation tables or HMI devices.
<b>EtherNet/IP protocol enabled</b>	Yes	TRUE/FALSE	TRUE	Allows you to enable or disable the EtherNet/IP protocol to connect to a network for data exchange.
<b>Modbus server enabled</b>	Yes	TRUE/FALSE	TRUE	Allows you to enable or disable the Modbus TCP server. As a consequence, this enables or disables access to memory objects %M and %MW using standard Modbus requests.
<b>Auto discovery protocol enabled</b>	Yes	TRUE/FALSE	TRUE	Allows you to enable or disable the auto discovery protocol to automatically detect devices on supported Ethernet fieldbuses.
<p>(1) You can select any one option for IP addressing. Selecting any one option disables the other options.</p> <p>(2) These options are enabled only if you select the option <b>Fixed IP Address</b> for IP addressing.</p> <p>(3) w, x, y, and z are the bytes that store the address and each byte can store a value in the range 0...255.</p>				

**NOTE:** When a protocol listed in **Security Parameters** is disabled, requests from the corresponding server type are ignored. The corresponding configuration screen remains accessible; however, program execution is not affected.

### Address Management

This diagram presents the different types of address systems for the M221 Logic Controller:



**NOTE:** If a device programmed to use the DHCP or BOOTP addressing methods is unable to contact its respective server, the controller uses the default IP address. It will, however, constantly repeat its request.

The IP process restarts in the following cases:

- Controller reboot
- Ethernet cable reconnection
- Application download (if IP parameters change)
- DHCP or BOOTP server detected after a prior addressing attempt was unsuccessful or when the DHCP address lease expires.

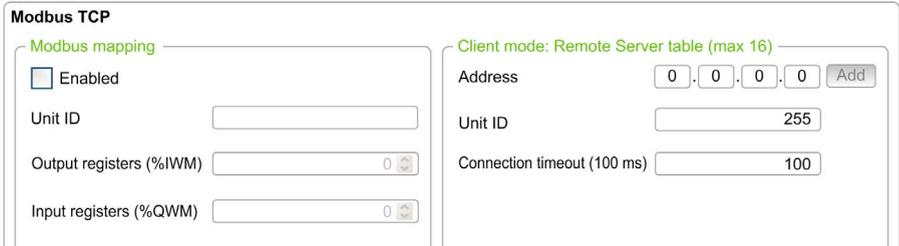
## Configuring Modbus TCP

### Introduction

You can configure the Ethernet port to enable the embedded Modbus TCP server giving the logic controller Modbus TCP abilities.

### Modbus TCP Configuration

This table describes how to configure Modbus TCP:

Step	Action
1	<p>Click the <b>Modbus TCP</b> node that appears below the <b>ETH1</b> node in the hardware tree to display the Modbus TCP properties</p> <p>This figure presents the properties displayed in the editor area:</p> 
2	<p>Select <b>Enabled</b> to edit the properties to configure the Modbus TCP protocol.</p> <p><b>NOTE:</b> If the <b>Enabled</b> button is grayed out, verify that the <b>Functional Level</b> of your application (<b>Programming</b> → <b>Tasks</b> → <b>Behavior</b> tab) is at least <b>Level 3.2</b>.</p> <p>For detailed information on Modbus TCP configuration parameters, refer to the table below.</p>

This table describes each parameter of the Modbus TCP configuration:

Parameter	Editable <sup>(1)</sup>	Value	Default Value	Description
<b>Modbus mapping</b>				
<b>Enabled</b>	Yes	TRUE/FALSE	FALSE	Select to enable the Modbus TCP configuration.  <b>NOTE:</b> If you deselect the <b>Enabled</b> check box, and you have used network variables in your program, they are no longer valid and your program can no longer be compiled. If you wish to temporarily disable the Modbus TCP/IP services without invalidating the use of its network variables, you can deactivate the <b>Security Parameters</b> for the protocol in the Ethernet properties window ( <a href="#">see page 121</a> ).
<b>Unit ID</b>	Yes	1...247	-	Specify the unit ID of the local server. Modbus TCP requests originating from a server with the same unit ID are sent to the Modbus mapping table instead of the regular Modbus server.
<b>Output registers (%IWM)</b>	Yes	1...20	10	The number of output registers available. Output registers are used to store the values of %IWM network objects ( <a href="#">see page 175</a> ).
<b>Input registers (%QWM)</b>	Yes	1...20	10	The number of Input registers available. Input registers are used to store the values of %QWM network objects ( <a href="#">see page 173</a> ).
<b>Client mode: Remote Server table (max 16)</b>				
<b>Address</b>	Yes	w.x.y.z <sup>(1)</sup>	0.0.0.0	Allows you to specify the IP address of the remote server. Also, refer to Adding Remote Servers ( <a href="#">see page 128</a> ).
<b>Unit ID</b>	Yes	0...255	255	Allows you to specify the unit ID of the remote server.
<b>Connection timeout (100 ms)</b>	Yes	0...65535	100	Allows you to specify the connection timeout duration.
<sup>(1)</sup> Only if the <b>Modbus server enabled</b> option is selected in the <b>Security Parameters</b> section of the Ethernet properties window ( <a href="#">see page 121</a> ).				

### Modbus TCP Slave Device I/O Mapping Table

When a Modbus TCP Slave device has been configured, Modbus commands sent to its unit ID (Modbus address) access network objects (%IWM and %QWM) of the controller, instead of the regular Modbus words accessed when the unit ID is 255. This facilitates read/write operations by a Modbus master I/O scanner application.

**NOTE:** If the `Unit ID` selected in the master is not the one configured in the M221 slave (and vice versa), data is read or written to %MWx instead of %IWMx and %QWMx. No Modbus error is returned.

The Modbus TCP Slave device responds to a subset of the Modbus function codes, but does so in a way that differs from Modbus standards, with the purpose of exchanging data with the external I/O scanner. The following Modbus function codes are supported by the Modbus TCP Slave device:

Function Code Dec (Hex)	Function	Comment
3 (3 hex)	Read output register	Allows the master I/O scanner to read network object %QWM of the device
4 (4 hex)	Read input registers	Allows the master I/O scanner to read network object %IWM of the device
6 (6 hex)	Write single register	Allows the master I/O scanner to write a single network object %IWM of the device
16 (10 hex)	Write multiple registers	Allows the master I/O scanner to write to multiple network objects %IWM of the device
23 (17 hex)	Read/write multiple registers	Allows the master I/O scanner to read network object %QWM and write network object %IWM of the device

### Adding Remote Servers

This table describes how to add a remote server for Modbus TCP:

Step	Action								
1	Enter the IP address in the <b>Address</b> field.								
2	Enter the value for <b>Unit ID</b> and <b>Connection timeout (100 ms)</b> .								
3	<p>Click the <b>Add</b> button.</p> <p><b>Result:</b> A list of remote servers that you have added appears on the screen. This figure presents the table listing the remote servers:</p> <table border="1" data-bbox="293 1268 917 1338"> <thead> <tr> <th>Index</th> <th>Address</th> <th>Unit ID</th> <th>Connection timeout (100 ms)</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>192.165.110.156</td> <td>255</td> <td>100</td> </tr> </tbody> </table>	Index	Address	Unit ID	Connection timeout (100 ms)	1	192.165.110.156	255	100
Index	Address	Unit ID	Connection timeout (100 ms)						
1	192.165.110.156	255	100						

This table describes each column of the table listing the remote servers:

Parameter	Editable	Value	Default Value	Description
<b>Index</b>	No	1...16	–	Displays the index number of the servers which are remotely connected.
<b>Address</b>	Yes	w.x.y.z <sup>(1)</sup>	0.0.0.0	Displays the IP address of the remote server.
<b>Unit ID</b>	Yes	0...255	255	Displays the unit ID of the remote server.
<b>Connection timeout (100 ms)</b>	Yes	0...65535	100	The connection timeout duration. The period of time (in units of 100 ms) during which the controller attempts to establish a TCP connection to the remote device. If at the end of this period a TCP connection is still not established, the controller stops connection attempts until the next connection request with an EXCH instruction.
<b>(1)</b> w, x, y, and z are the bytes that store the address and each byte can store a value in the range 0...255.				

Click the close button in the row to remove a remote server.

Double-click the remote server entry in a row to edit the values.

## Configuring EtherNet/IP

### Introduction

This section describes the configuration of the EtherNet/IP connection to the controller.

For further information about EtherNet/IP, refer to [www.odva.org](http://www.odva.org)

### EtherNet/IP Adapter Configuration

The following table describes how to display the EtherNet/IP Adapter configuration window:

Step	Action
1	<p>Click the <b>EtherNet/IP adapter</b> node that appears below the <b>ETH1</b> node in the hardware tree. This figure presents the properties of the EtherNet/IP Adapter in the editor area:</p> 
2	<p>Select <b>Enabled</b> to edit the properties to configure the EtherNet/IP Adapter.</p> <p><b>NOTE:</b> If the <b>Enabled</b> button is grayed out, verify that the <b>Functional Level</b> of your application (<b>Programming</b> → <b>Tasks</b> → <b>Behavior</b> tab) is at least <b>Level 3.2</b>.</p> <p>For detailed information on the EtherNet/IP Adapter configuration parameters, refer to the table below.</p>
3	<p>Click <b>Apply</b>.</p>

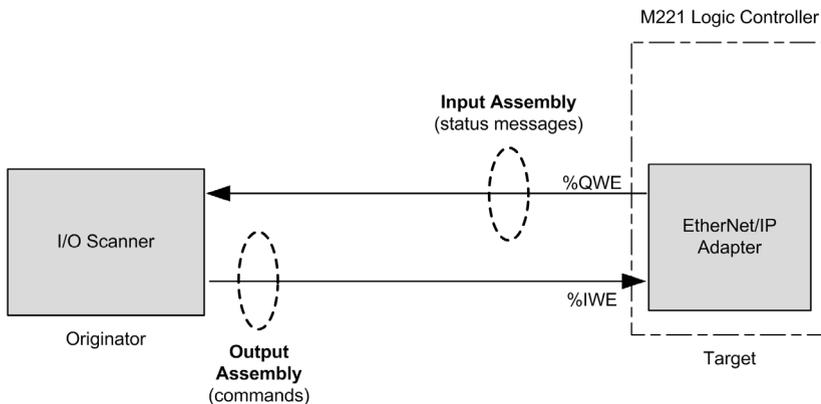
### EtherNet/IP Adapter Properties

This table describes each parameter of the EtherNet/IP Adapter configuration:

Parameter	Editable	Value	Default Value	Description
<b>Enabled</b>	Yes	TRUE/FALSE	FALSE	Select to enable the EtherNet/IP Adapter configuration.  <b>NOTE:</b> If you deselect the <b>Enabled</b> check box, and you have used network variables in your program, they are no longer valid and your program can no longer be compiled. If you wish to temporarily disable the EtherNet/IP Adapter services without invalidating the use of its network variables, you can deactivate the <b>Security Parameters</b> for the protocol in the Ethernet properties window ( <i>see page 121</i> ). When disabled, by deselecting the <b>Enabled</b> checkbox, configured fallback values ( <i>see page 170</i> ) of %QWE objects, as well as symbols and comments, are lost.
<b>Input assembly (Target --&gt;Originator, %QWE)</b>				
<b>Instance</b>	Yes	1...255	100	The identifier of the Input assembly.
<b>Size (words)</b>	Yes	1...20	20	The size of the Input assembly.
<b>Output assembly (Originator--&gt;Target, %IWE)</b>				
<b>Instance</b>	Yes	1...255	150	The identifier of the Output assembly.
<b>Size (words)</b>	Yes	1...20	20	The size of the Output assembly.

**NOTE:** Output means output from the Scanner controller (%IWE for the Adapter).  
Input means input from the Scanner controller (%QWE for the Adapter).

The following graphic presents the directionality of Input assembly and Output assembly in EtherNet/IP communications:



### EDS File

A template electronic data sheet (EDS) file, **M221\_EDS\_Model.eds**, is provided in *SoMachine Basic installation folder \Firmwares & PostConfiguration*.

Modify the file as described in the user guide to be found in the same folder.

---

## Section 6.2

### Serial Line Configuration

---

#### Configuring Serial Line

##### Introduction

The M221 Logic Controller references are equipped with at least 1 serial line. The controller references without the Ethernet feature support 2 serial lines:

- SL1 (serial line).
- SL2 (serial line).

The SL1 port supports these features to ease modem connection:

- An Init command to send to the modem its configuration. This command is automatically sent by the controller after an application download or at power on.
- %S105 to send per application the Init command to the modem.
- %SW167 to provide a status on the Init command operation.

SL2 does not support these features dedicated to modem.

Modem connection allows you to perform:

- Remote access to the controller for the purpose of programming and/or monitoring. In this case, a local modem must be connected to the PC running the SoMachine Basic software, and a modem connection must be configured.
- Exchanges between the controllers using Modbus protocol.
- Send or receive messages with any device using `SEND_RECEIVE_MSG` function block.
- Send or receive SMS on a mobile phone or on other devices able to send or receive SMS.

Each serial line can be configured for any one of the following protocols:

- Modbus
- ASCII
- TMH2GDB

You can configure both protocol and serial line settings. Serial lines are configured for the Modbus RTU protocol by default.

## Serial Line Configuration

This table describes how to configure the serial line:

Step	Action
1	<p>Click the <b>SL1 (Serial line)</b> or <b>SL2 (Serial line)</b> node in the hardware tree to display the serial line configuration.</p> <div data-bbox="293 344 1141 912" style="border: 1px solid #ccc; padding: 10px;"> <p><b>Serial line configuration</b></p> <p><b>Protocol Settings</b></p> <p>Protocol <span style="border: 1px solid #ccc; padding: 2px;">Modbus ▾</span></p> <hr/> <p><b>Serial line settings</b></p> <p>Baud rate <span style="border: 1px solid #ccc; padding: 2px;">19200 ▾</span></p> <p>Parity <span style="border: 1px solid #ccc; padding: 2px;">Even ▾</span></p> <p>Data bits 8</p> <p>Stop bits <span style="border: 1px solid #ccc; padding: 2px;">1 ▾</span></p> <p>Physical medium</p> <p><input checked="" type="radio"/> RS-485      Polarization    No</p> <p><input type="radio"/> RS-232</p> <p style="text-align: right;"><span style="border: 1px solid #ccc; padding: 2px 10px;">Apply</span> <span style="border: 1px solid #ccc; padding: 2px 10px;">Cancel</span></p> </div>

This table describes the configuration parameters of the serial line:

Parameter	Editable	Value	Default Value	Description
<b>Protocol Settings</b>				
<b>Protocol</b>	Yes	<b>Modbus ASCII TMH2GDB</b>	<b>Modbus</b>	Select a protocol from the drop-down list.  <b>NOTE:</b> When using an <b>SR2MOD03</b> and the function block SMS, select <b>ASCII</b> protocol.
<b>Serial Line Settings</b>				
<b>Baud rate</b>	Yes	<b>1200 2400 4800 9600 19200 38400 57600 115200</b>	<b>19200</b>	Allows you to select the data transmission rate (bits per second) from the drop-down list.
<b>Parity</b>	Yes	<b>None Even Odd</b>	<b>Even</b>	Allows you to select the parity of the transmitted data for error detection. Parity is a method of error detection in transmission. When parity is used with a serial port, an extra data bit is sent with each data character, arranged so that the number of bits set to 1 in each character, including the parity bit, is always odd or always even. If a byte is received with an incorrect number of bits set to 1, the byte is invalid.
<b>Data bits</b>	Yes (only for the <b>ASCII</b> protocol)	<b>7 8</b>	<b>8</b>	Allows you to select the data bit from the drop-down list. The number of data bits in each character can be 7 (for true ASCII) or 8.
<b>Stop bits</b>	Yes	<b>1 2</b>	<b>1</b>	Allows you to select the stop bit from the drop-down list. Stop bit is a bit indicating the end of a byte of data. For electronic devices usually 1 stop bit is used. For slow devices like electromechanical teleprinters, 2 stop bits are used.

Parameter	Editable	Value	Default Value	Description
<b>Physical medium</b>	Yes	<b>RS-485</b> TRUE/FALSE  <b>RS-232</b> TRUE/FALSE	<b>RS-485</b> TRUE	Allows you to select the physical medium for communication. You can select either <b>RS-485</b> or <b>RS-232</b> medium for serial line 1. For serial line 2, only <b>RS-485</b> medium is available. A physical medium in data communications is the transmission path over which a signal propagates. It is an interface for interconnection of devices with the logic controller. <b>NOTE:</b> When using an <b>SR2MOD03</b> , select the <b>RS-232</b> option.
<b>Polarization</b>	Yes (for cartridges only) No (for the controller)	<b>Yes</b> <b>No</b>	<b>No</b>	Polarization resistors are integrated in the cartridge module. For the controller, this parameter is disabled and for the cartridges, this parameter allows you to switch on or off polarization.

## Device Configuration

This table describes each parameter of the selected device:

Parameter	Editable	Value	Default Value	Description
<b>Device Settings</b>				
<b>Device</b> (inactive for <b>Display</b> node)	Yes	<b>None</b> <b>Generic Modem</b> <b>SR2MOD01</b> <b>SR2MOD03</b>	<b>None</b>	Select a device from the drop-down list. Select <b>SR2MOD03</b> to use the %SEND_RECV_SMS function block. <b>TMH2GDB</b> is displayed when the <b>Display</b> node is selected.
<b>Init command</b> (only for SL1)	Yes	-	-	Init command is a set of HAYES commands sent to the modem connected on the serial line 1. It is an ASCII string limited to 128 characters. The logic controller uses this string to configure and verify the modem. The Init command is sent to the modem: <ul style="list-style-type: none"> <li>• at Power on.</li> <li>• when application is downloaded to the controller.</li> <li>• when the %S105 system bit is set.</li> </ul> %SW167 provides the status of the initialization command sent to the modem. Init command is initialized by SoMachine Basic for <b>SR2MOD03</b> modem. For more information, refer to the <i>SR2MOD02 and SR2MOD03 Wireless Modem User Guide</i> (see page 10). <b>NOTE:</b> To use SMS function block, change the default Init command to: AT&F;E0;S0=2;Q0;V1;+WIND=0;+CBST=0,0,1;&W;+CNMI=0,2,0,0,0;+CSAS;+CMGF=0;+CMEE=1

Parameter	Editable	Value	Default Value	Description
<b>Protocol Settings for a Modbus device</b>				
<b>Transmission mode</b>	Yes	<b>RTU</b> <b>ASCII</b>	<b>RTU</b>	Allows you to select the protocol transmission mode for communication from the drop-down list. Select <b>ASCII</b> to use the %SEND_RECV_SMS function block. Protocol advanced parameters are displayed based on the selected protocol. Refer to the following figures and tables.
<b>Addressing</b>	Yes	<b>Slave</b> TRUE/FALSE  <b>Master</b> TRUE/FALSE	<b>Slave</b> TRUE	Allows you to select the addressing mode. You can only select either of the <b>Slave</b> or <b>Master</b> addressing. Enabling any of the addressing mode, disables the other one.
<b>Address [1...247]</b>	Yes	1...247	1	Allows you to specify the address ID of the slave. <b>NOTE:</b> This field is displayed only for the addressing of the slave. For master, this field does not appear on the screen.
<b>Response timeout (x 100 ms)</b>	Yes	0...255	10	Defines the maximum time that the controller waits for a response before terminating the exchange in error. Enter 0 to disable the timeout.
<b>Time between frames (ms)</b>	Yes	1...255	10	The period of time between frames (corresponds to inter-frame delay used in other products). <b>NOTE:</b> The value is subject to adjustment to conform to Modbus standard 3.5 character time delay.
<b>Protocol Settings for an ASCII device</b>				
<b>Response timeout (x 100 ms)</b>	Yes	0...255	10	Defines the maximum time that the controller waits for a response before terminating the exchange in error. Enter 0 to disable the timeout. <b>NOTE:</b> When using an <b>SR2MOD03</b> and the function block SMS, enter 0 to disable the timeout.
<b>Stop condition</b>				

Parameter	Editable	Value	Default Value	Description
<b>Frame length received</b>	Yes (only if the check box is selected)	1...255	0 (if check box is not selected) 1 (if check box is selected)	Allows you to specify the length of the received frame. <b>NOTE:</b> You can configure only one parameter for stop condition that is either <b>Frame length received</b> or <b>Frame received timeout (ms)</b> .
<b>Frame received timeout (ms)</b>	Yes (only if the check box is selected)	1...255	0 (if check box is not selected) 10 (if check box is selected)	Allows you to specify the timeout duration for the received frame. <b>NOTE:</b> When using an <b>SR2MOD03</b> and the function block SMS, select the checkbox and enter 200.
<b>Frame structure</b>				
<b>Start character</b>	Yes (only if the check box is selected)	1...255	0 (if check box is not selected) 58 (if check box is selected)	Allows you to specify the start character of the frame. The ASCII character corresponding to the start character value is displayed on right-hand side of the value field.
<b>First end character</b>	Yes	1...255	0 (if check box is not selected) 10 (if check box is selected)	Allows you to specify the first end character of the frame. <b>NOTE:</b> To be able to disable the <b>First end character</b> , configure at least one stop condition parameter. The ASCII character corresponding to the first end character value is displayed on right-hand side of the value field.
<b>Second end character</b>	Yes (only if the check box is selected)	1...255	0 (if check box is not selected) 10 (if check box is selected)	Allows you to specify the second end character of the frame. <b>NOTE:</b> This field is disabled when the disabled <b>First end character</b> is disabled. The ASCII character corresponding to the second end character value is displayed on right-hand side of the value field.
<b>Send frame characters</b>	Yes	TRUE/FALSE	FALSE	Allows you to enable or disable the automatic addition of start, first end, and second end characters (when defined) in the frames sent.

Parameter	Editable	Value	Default Value	Description
<b>Protocol Settings for a Display device</b>				
<b>Time between frames (ms)</b>	Yes	1...255	10	The period of time between frames (corresponds to inter-frame delay used in other products).  <b>NOTE:</b> The value is subject to adjustment to conform to Modbus standard 3.5 character time delay.

### Configure the Modem Init Command

The Init command is a set of Hayes commands to operate with a modem. The default Init command provided by SoMachine Basic configuration screen is to be used with a modem to match with the default serial line configuration for remote access, exchanges between controllers or sending/receiving messages.

To send or receive SMS, the Init command must be modified:

```
AT&F;E0;S0=2;Q0;V1;+WIND=0;+CBST=0,0,1;&W;+CNMI=0,2,0,0,0;+CSAS;+CMGF=0;+CMEE=1
```

Use a PC terminal software if you need to adapt the Init command.

### SR2MOD01 Hayes Command

The default Init command provided by SoMachine Basic is:

```
ate0\n0\v1&d0&k0s0=1s89=0$EB0#p0$sb19200n0s28=1s37=13&w0
```

### SR2MOD03 Hayes Command

The default Init command provided by SoMachine Basic is:

```
AT&F;E0;S0=2;Q0;V1;+WIND=0;+CBST=0,0,1;&W;+CMGF=1;+CNMI=0,2,0,0,0;+CSAS
```

To send or receive SMS, the command must be modified:

```
AT&F;E0;S0=2;Q0;V1;+WIND=0;+CBST=0,0,1;&W;+CNMI=0,2,0,0,0;+CSAS;+CMGF=0;+CMEE=1
```

## Section 6.3

### Supported Modbus Function Codes

#### Supported Modbus Function Codes

##### Presentation

This section lists the function codes supported by both serial Modbus and Modbus TCP and their effect on controller memory variables

##### Modbus Slave

The following Modbus requests are supported:

Supported Modbus Function Code Dec (Hex)	Supported Sub-Function Code	Description
1 (1 hex) or 2 (2 hex)	–	Read multiple internal bits %M
3 (3 hex) or 4 (4 hex)	–	Read multiple internal registers %MW
5 (5 hex)	–	Write single internal bit %M
6 (6 hex)	–	Write single internal register %MW
8 (8 hex)	0 (0 hex), 10 (0A hex)...18 (12 hex)	Diagnostics
15 (0F hex)	–	Write multiple internal bits %M
16 (10 hex)	–	Write multiple internal registers %MW
23 (17 hex)	–	Read/write multiple internal registers %MW
43 (2B hex)	14 (0E hex)	Read device identification (regular service)

##### NOTE:

The impact of Modbus function codes used by a master M221 Logic Controller depends on the slave device type. In the major types of slave device:

- Internal bit means %M
- Input bit means %I
- Internal register means %MW
- Input register means %IW

Depending on the type of slave and the slave address, an internal bit should be a %M or %Q; an input bit should be a %I or %S, an input register should be a %IW or a %SW and an internal register should be a %MW or a %QW.

For more details, refer to the documentation of the slave device.

## Modbus Mapping Table

Modbus TCP slave devices support a subset of the Modbus function codes. Function codes originating from a Modbus master with matching unit ID are directed to the Modbus mapping table and access network objects (%IWM and %QWM) of the controller. Refer to Modbus TCP Slave Device I/O Mapping Table ([see page 128](#)).

---

# Chapter 7

## SD Card

---

### Introduction

The Modicon M221 Logic Controller allows file transfers with an SD card.

This chapter describes how to manage Modicon M221 Logic Controller files with an SD card.

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
File Management Operations	144
SD Card Supported File Types	146
Clone Management	148
Firmware Management	150
Application Management	152
Post Configuration Management	154
Error Log Management	156
Memory Management: Backing Up and Restoring Controller Memory	159

## File Management Operations

### Introduction

The Modicon M221 Logic Controller allows the following types of file management with an SD card:

- Clone management (*see page 148*): back up the application, firmware, and post configuration (if it exists) of the logic controller
- Firmware management (*see page 150*): download firmware directly to the logic controller, and load firmware to the Remote Graphic Display
- Application management (*see page 152*): back up and restore the logic controller application, or copy it to another logic controller of the same reference
- Post configuration management (*see page 154*): add, change, or delete the post configuration file of the logic controller
- Error log management (*see page 156*): back up or delete the error log file of the logic controller
- Memory management (*see page 159*): Back up and restore memory objects of the controller

#### NOTE:

- Logic controller logic solving and services execution continues during file transfers.
- Certain commands require a power cycle of the logic controller. See the description of the commands for more information.
- The Modicon M221 Logic Controller accepts only SD cards formatted in FAT or FAT32.

With the use of the SD card, powerful operations can be automatically conducted affecting the behavior of your logic controller and resident application. Care must be taken when inserting an SD card into the controller; you must be aware of the affect that the contents of the SD card will have on your logic controller.

**NOTE:** File management with SD card uses script files. These scripts can be automatically created with the **Memory Management** task (*see SoMachine Basic, Operating Guide*).

## WARNING

### UNINTENDED EQUIPMENT OPERATION

- You must have operational knowledge of your machine or process before connecting an SD card to your logic controller.
- Ensure that guards are in place so that any potential affect of the contents of the SD card will not cause injury to personnel or damage to equipment.

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

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.**

## SD Card Supported File Types

### Introduction

This table lists the file locations and types of file that can be managed:

SD card folder	Description	Default file name
/	Script file	Script.cmd
/	Script log	Script.log
/sys/os	Logic controller firmware file	M221.mfw
/disp/	Remote Graphic Display firmware file	TMH2GDB.mfw
/usr/app	Application file	*.smbk
/usr/cfg	Post configuration file	Machine.cfg
/usr/mem	Memory back up file	Memories.csv
/sys/log	Detected error log file	PlcLog.csv

### Script File Commands

A script file is a text file stored in the root directory of the SD card containing commands to manage exchanges with the controller.

This table describes the supported script commands:

Command	Description
<b>Download</b>	Download a file from the SD card to the controller.
<b>Upload</b>	Upload files contained in controller memory to the SD card.
<b>Delete</b>	Delete files contained in a controller.

### Script File Examples

**Download** commands:

```
Download "/usr/cfg"
```

```
Download "/sys/os/M221.mfw"
```

```
Download "/disp/TMH2GDB.mfw"
```

**Upload** commands:

```
Upload "/usr/app/*"
```

```
Upload "/usr/cfg/Machine.cfg"
```

**Delete** commands:

```
Delete "/usr/app/*"
```

```
Delete "/sys/log/PlcLog.csv"
```

**Script Log**

A `script.log` file is automatically created in the SD card root directory after script operations. The status of the script operations can be verified reading this file.

## Clone Management

### Cloning

Cloning allows you to automatically back up the application, firmware, and post configuration (if it exists) of the Modicon M221 Logic Controller to the SD card.

The SD card can then be used to later restore the firmware, application, and post configuration (if it exists) to the logic controller, or copy them to another logic controller with the same reference.

Before cloning a controller, the M221 Logic Controller verifies whether the application is copy-protected or not. For details, refer to Password Protecting an Application (see *SoMachine Basic, Operating Guide*).

#### NOTE:

- The SD card must be empty and correctly formatted to perform this procedure.
- The detected error log and data memory are not cloned.
- If the application is password-protected, the clone operation is blocked (the **SD** LED is flashing).

### Creating a Clone SD Card

This procedure describes how to copy the application, firmware, and post configuration (if it exists) from the controller to an SD card:

Step	Action
1	Format an SD card on the PC.
2	Insert the SD card into the controller. <b>Result:</b> The clone operation starts automatically and the <b>SD</b> LED turns ON.
3	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file. <b>NOTE:</b> The clone operation lasts 2 or 3 minutes. The clone operation has a low priority in order to minimize impact on the user logic and communication performance of the logic controller. Depending on the amount of free time in your program, the operation may require more time to complete if the logic controller is in <code>RUNNING</code> state compared to the <code>STOPPED</code> state.
4	Remove the SD card from the controller.

## Restoring or Copying from a Clone SD Card

This procedure describes how to download the application, firmware, and post configuration (if it exists) stored in the SD card to your controller:

Step	Action
1	Remove power from the controller.
2	Insert the SD card into the controller.
3	Restore power to the controller. <b>Result:</b> The clone operation is in progress. <b>NOTE:</b> The <b>SD</b> LED is turned on during the operation.
4	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file.
5	Remove the SD card to restart the controller.

**NOTE:** Downloading a cloned application to the controller first removes the existing application from controller memory, regardless of any user access-rights that may be enabled in the target controller.

## Firmware Management

### Overview

You can use an SD card to download firmware updates directly to the logic controller or a Remote Graphic Display.

Refer to Controller States and Behavior ([see page 47](#)) for information on the logic controller operating states and status of the LEDs.

### Downloading Firmware to the Controller

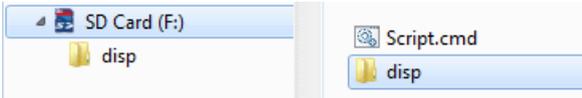
This table describes how to download a firmware to the logic controller using an SD card:

Step	Action
1	Remove power from the controller.
2	Insert an empty SD card into the PC that is running SoMachine Basic.
3	Create a file called <code>script.cmd</code> in the SD card root directory.
4	Edit the file and insert the following command: <code>Download "/sys/os"</code>
5	Create the folder path <code>\sys\os</code> in the SD card root directory and copy the firmware file in the <code>os</code> folder:  <p><b>NOTE:</b> A firmware file example and the script are available in the folder <code>Firmwares &amp; PostConfiguration\M221\</code> of the SoMachine Basic installation folder. The firmware file name for the M221 Logic Controller is <code>M221.mfw</code>.</p>
6	Remove the SD card from the PC and insert it into the SD card slot of the logic controller.
7	Restore power to the controller. <b>Result:</b> Copying of the firmware file begins. During the operation, the <b>SD</b> system LED on the logic controller is ON. <b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress.
8	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file.
9	Remove the SD card.
10	Reconnect the USB programming cable to the logic controller and login to the logic controller with the SoMachine Basic software.

## Downloading Firmware to the Remote Graphic Display

**NOTE:** Before downloading, verify whether the firmware version to be installed is compatible with the installed SoMachine Basic software version and the logic controller firmware version. Refer to Compatibility of the Remote Graphic Display (see *Modicon TMH2GDB, Remote Graphic Display, User Guide*)

This table describes how to download a firmware to the Remote Graphic Display using an SD card:

Step	Action
1	Connect the Remote Graphic Display to the logic controller (see <i>Modicon TMH2GDB, Remote Graphic Display, User Guide</i> ).
2	Insert an empty SD card into the PC that is running SoMachine Basic.
3	Create a file called <code>script.cmd</code> in the SD card root directory.
4	Edit the file and insert the following command: Download <code>"/disp/TMH2GDB.mfw"</code>
5	<p>Create the folder path <code>/disp/</code> in the SD card root directory and copy the firmware file in the <code>disp</code> folder:</p>  <p><b>NOTE:</b> The firmware file and an example script are available in the folder <code>Firmwares &amp; PostConfiguration\TMH2GDB\</code> of the SoMachine Basic installation folder. The firmware file name for the Remote Graphic Display is <code>TMH2GDB.mfw</code>.</p>
6	<p>Remove the SD card from the PC and insert it into the SD card slot of the M221 Logic Controller.</p> <p><b>Result:</b> The logic controller begins transferring the firmware file from the SD card to the Remote Graphic Display. During this operation:</p> <ul style="list-style-type: none"> <li>● the message <b>File Transfer</b> is displayed on the Remote Graphic Display</li> <li>● the <b>SD</b> system LED on the M221 Logic Controller is turned on</li> <li>● system word <code>%SW182</code> is set to 5 (<b>Display firmware transfer in progress</b>)</li> </ul> <p><b>NOTE:</b> Do not disconnect the Remote Graphic Display or remove power from M221 Logic Controller while the operation is in progress.</p>
7	<p>Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file.</p> <p><b>NOTE:</b> Restoring the file system on the Remote Graphic Display (red backlight) is part of the process.</p>

## Application Management

### Overview

You can use an SD card to back up and restore your controller application, or copy it to another controller with the same reference.

### Backing Up an Application

This table describes how to back up the controller application on the SD card:

Step	Action
1	Create a <code>script.cmd</code> file with a text editor on your PC.
2	Edit the file and insert the following line: <code>Upload "/usr/app"</code>
3	Copy the script file to the root folder of the SD card.
4	<p>Insert the prepared SD card in the controller.</p> <p><b>Result:</b> Copying of the application file begins. During the operation, the <b>SD</b> system LED on the logic controller is <b>ON</b>.</p> <p><b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress.</p> <p><b>NOTE:</b> The application backup process has a low priority in order to minimize impact on the program and communication performance of the logic controller. Depending on the amount of free time in your program, the operation may take considerably longer to complete if the logic controller is in <b>RUNNING</b> state compared to the <b>STOPPED</b> state.</p>
5	<p>Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing).</p> <p>If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file.</p> <p><b>Result:</b> The application file (<code>*.smbk</code>) is saved on the SD card.</p>

## Restoring an Application or Copying an Application to Another Controller

This table describes how to transfer the controller application from the SD card to the controller:

Step	Action
1	Take an SD card previously created and edit the <code>script.cmd</code> file in the root folder of the SD card with a text editor.
2	Replace the content of the script by the following line: <code>Download "/usr/app"</code>
3	Remove power from the controller.
4	Insert the prepared SD card in the controller.
5	Restore power to the controller. <b>Result:</b> Copying of the application file begins. During the operation, the <b>SD</b> system LED on the logic controller is ON. <b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress.
6	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file.
7	Remove the SD card to restart the controller.

## Post Configuration Management

### Overview

You can use an SD card to add, change, or delete the post configuration file of your controller.

### Adding or Changing a Post Configuration

This table describes how to add or change the controller post configuration:

Step	Action
1	Create a file called <code>script.cmd</code> .
2	Edit the file and insert the following line: Download <code>"/usr/cfg"</code>
3	Copy the post configuration file ( <code>Machine.cfg</code> ) to the folder <code>\usr\cfg</code> and the script file to the root folder of the SD card:  <b>NOTE:</b> A post configuration file example and the associated script are available in the directory <code>Firmwares &amp; PostConfiguration\PostConfiguration\add_change\</code> of the SoMachine Basic installation directory.
4	If necessary, edit the <code>Machine.cfg</code> file to configure your post configuration parameters.
5	Insert the prepared SD card in the controller. <b>Result:</b> Downloading of the post configuration file begins. During the operation, the <b>SD</b> system LED on the logic controller is ON. <b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress. <b>NOTE:</b> Before the download the file format is verified, as well as if all of the channels, parameters, and values configured are valid; in case of detected error the download is aborted.
6	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file.
7	Do a power cycle or an initialization command to apply the new post configuration file.

## Reading a Post Configuration File

This table describes how to read the post configuration file of the controller:

Step	Action
1	Create a <code>script.cmd</code> file with a text editor on your PC.
2	Edit the file and insert the following line: <code>Upload "/usr/cfg"</code>
3	Copy the script file to the root folder of the SD card.
4	Insert the prepared SD card in the controller. <b>Result:</b> Copying of the post configuration file begins. During the operation, the <b>SD</b> system LED on the logic controller is ON. <b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress. <b>NOTE:</b> The application backup process has a low priority to minimize impact on the program and communication performance of the logic controller. Depending on the amount of free time in your program, the operation may take considerably longer to complete, if the logic controller is in <b>RUNNING</b> state compared to the <b>STOPPED</b> state.
5	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file. <b>Result:</b> The post configuration file is saved on the SD card.

## Removing a Post Configuration File

This table describes how to remove the post configuration file of the controller:

Step	Action
1	Insert an empty SD card into the PC that is running SoMachine Basic.
2	Create a file called <code>script.cmd</code> .
3	Edit the file and insert the following line: <code>Delete "/usr/cfg"</code>
4	Copy the script file available in the directory <code>Firmwares &amp; PostConfiguration\PostConfiguration\remove\</code> of the SoMachine Basic installation directory to the root directory of the SD card.
5	Insert the prepared SD card in the controller. <b>Result:</b> The post configuration file is removed. During the operation, the <b>SD</b> system LED on the logic controller is ON. <b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress.
6	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file.
7	Do a power cycle or an initialization command to apply the application parameters.

## Error Log Management

### Overview

You can use the SD card to back up or delete the error log file of the logic controller.

### Backing Up the Error Log

This table describes how to back up the logic controller error log file on the SD card:

Step	Action
1	Create a <code>script.cmd</code> file with a text editor on your PC.
2	Edit the file and insert the following line: <code>Upload "/sys/log"</code>
3	Copy the script file to the root folder of the SD card.
4	Insert the prepared SD card in the logic controller. <b>Result:</b> Transfer of the error log file begins. During the operation, the <b>SD</b> system LED on the logic controller is ON. <b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress.
5	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file. <b>Result:</b> The error log file ( <code>PlcLog.csv</code> ) is saved on the SD card.

### Deleting the Error Log

This table describes how to delete the error log file in the logic controller:

Step	Action
1	Create a <code>script.cmd</code> file with a text editor on your PC.
2	Edit the file and insert the following line: <code>Delete "/sys/log"</code>
3	Copy the script file to the root folder of the SD card.
4	Insert the prepared SD card in the logic controller. <b>Result:</b> Deleting of the error log file begins. During the operation, the <b>SD</b> system LED on the logic controller is ON. <b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress.
5	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file. <b>Result:</b> The error log file ( <code>PlcLog.csv</code> ) is deleted from the logic controller.

## Error Log Format

The logic controller provides an error list containing the last 10 detected errors in the log memory. Each error entry into the error log file is composed of four parts:

- Date and time
- Level
- Context
- Error code

After an upload through the SD card, the code is represented as in the example below:

```
02/06/14, 12:04:01, 0x01110001
```

This table describes the meaning of the hexadecimal error representation:

Group	Hexadecimal error code	Error Description	Result
General	08000011	Invalid hardware calibration parameters	Ethernet channel is inoperative %SW118.bit10 set to 0 <b>ERR LED flashes</b>
Operating system	0F01xxxx	Operating system error detected	Transition to HALTED state
Memory management	0F030009	Internal memory allocation error detected	Transition to HALTED state
Watchdog timer	0104000A	Logic controller resource utilization greater than 80% - first detection	Watchdog timeout signaled: %S11 set to 1 <b>ERR LED flashes</b>
	0804000B	Logic controller resource utilization greater than 80% - second consecutive detection	Transition to HALTED state
	0804000C	Task watchdog timer in master task	Transition to HALTED state
	0804000D	Task watchdog timer in periodic task	Transition to HALTED state
Battery	0105000E	Battery is depleted	Depleted battery signaled: %S75 set to 1 <b>BAT LED illuminated</b>
RTC	01060012	RTC is invalid	Invalid RTC signaled: %SW118.bit12 set to zero %S51 set to 1
User application	0807000F	Application is not compatible with firmware	Transition to EMPTY state
	08070010	Checksum error detected	Transition to EMPTY state
Ethernet	010B0014	Duplicate IP address detected	Duplicate IP signaled: %SW62 set to 1 %SW118.bit9 set to 0 <b>ERR LED flashes</b>

Group	Hexadecimal error code	Error Description	Result
Embedded I/O	010D0013	Short-circuit detected on protected output	Over-current signaled: %SW139 set to 1 (depending on the output block) <b>ERR</b> LED flashes
Read non-volatile memory	01110000	Read error detected - file not found	Unsuccessful read operation
	01110001	Read error detected - incorrect logic controller type	
	01110002	Read error detected - incorrect header	
	01110003	Read error detected - incorrect area descriptor	
	01110004	Read error detected - incorrect area descriptor size	
Write non-volatile memory	01120002	Write error detected - incorrect header	Unsuccessful write operation
	01120004	Write error detected - incorrect area descriptor size	
	01120005	Write error detected - unsuccessful erase	
	01120006	Write error detected - incorrect header size	
Persistent variable	01130007	Checksum error detected in persistent variables	Persistent variables cannot be restored
	01130008	Size error detected in persistent variables	
Ethernet IP	01140012	Unsuccessful Ethernet IP variable creation	Variable cannot be created, unsuccessful operation

## Memory Management: Backing Up and Restoring Controller Memory

### Overview

You can use an SD card to back up and restore controller memory objects, or copy the memory objects to another controller.

### Backing Up Controller Memory

Step	Action
1	Create a <code>script.cmd</code> file with a text editor on your PC.
2	Edit the file and insert the following line: <code>Upload "/usr/mem"</code>
3	Copy the script file to the root folder of the SD card.
4	Insert the prepared SD card in the controller. <b>Result:</b> Copying of the memory begins. During the operation, the <b>SD</b> system LED on the logic controller is ON. <b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress. <b>NOTE:</b> The memory backup process has a low priority to minimize impact on the program and communication performance of the logic controller. Depending on the amount of free time in your program, the operation may take considerably longer to complete if the logic controller is in <b>RUNNING</b> state compared to the <b>STOPPED</b> state.
5	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file. <b>Result:</b> The memory file (*.csv) is saved on the SD card.

### Restoring Controller Memory or Copying to Another Controller

Step	Action
1	Edit the <code>script.cmd</code> file in the root folder of the SD card with a text editor.
2	Replace the content of the script by the following line: <code>Download "/usr/mem"</code>
3	Insert the prepared SD card in the controller. <b>Result:</b> Copying of the memory file begins. During the operation, the <b>SD</b> system LED on the logic controller is ON. <b>NOTE:</b> Avoid removing power from the logic controller while the operation is in progress.
4	Wait until the end of the operation (until the <b>SD</b> LED is turned OFF or flashing). If an error is detected, the <b>SD</b> and <b>ERR</b> LEDs flash and the detected error is logged in <code>Script.log</code> file.



---

# Part III

## Programming the M221 Logic Controller

---

### Overview

This part provides information about the system and I/O objects specific to the M221 Logic Controller. These objects are displayed in the **Programming** tab.

For descriptions of all other objects, refer to SoMachine Basic Generic Functions Library Guide.

### What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
8	I/O Objects	163
9	Network Objects	169
10	Function Blocks	177
11	System Objects	179



---

# Chapter 8

## I/O Objects

---

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Digital Inputs (%I)	164
Digital Outputs (%Q)	165
Analog Inputs (%IW)	166
Analog Outputs (%QW)	168

## Digital Inputs (%I)

### Introduction

Digital input bit objects are the image of digital inputs on the logic controller.

### Displaying Digital Input Properties

Follow these steps to display properties of the digital inputs:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>I/O objects</b> → <b>Digital inputs</b> . <b>Result:</b> Digital input properties appear on the screen.

### Digital Inputs Properties

This table describes each property of the digital input:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	True/False	False	Indicates whether the input channel is being referenced in a program.
<b>Address</b>	No	%I0.i	–	Displays the address of the digital input on the controller, where i represents the channel number. If the controller has n digital input channels, the value of i is given as 0...n-1. For example, %I0.2 is the digital input at the digital input channel number 2 of the logic controller.
<b>Symbol</b>	Yes	–	–	The symbol associated with this address. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with this input. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of this symbol throughout the program and/or program comments.
<b>Comment</b>	Yes	–	–	A comment associated with this address. Double-click in the <b>Comment</b> column and type an optional comment to associate with this channel.

## Digital Outputs (%Q)

### Introduction

Digital output bit objects are the image of digital outputs on the logic controller.

### Displaying Digital Output Properties

Follow these steps to display properties of the digital outputs:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>I/O objects</b> → <b>Digital outputs</b> . <b>Result:</b> Digital output properties appear on the screen.

### Digital Outputs Properties

This table describes each property of the digital output:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	True/False	False	Indicates whether the output channel is being referenced in a program.
<b>Address</b>	No	%Q0.i	–	Displays the address of the digital output on the controller, where i represents the channel number. If the controller has n digital output channels, the value of i is given as 0...n-1. For example, %Q0.3 is the digital output at the digital output channel number 3 of the logic controller.
<b>Symbol</b>	Yes	–	–	The symbol associated with this address. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with this output. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of this symbol throughout the program and/or program comments.
<b>Comment</b>	Yes	–	–	The comment associated with this address. Double-click in the <b>Comment</b> column and type an optional comment to associate with this channel.

## Analog Inputs (%IW)

### Introduction

Analog input word objects are the digital values of an analog signal connected to the logic controller.

Two 0-10V analog inputs are embedded in the logic controller. The embedded analog inputs use a 10 bits resolution converter so that each increment is approximately 10 mV ( $10V/2^{10}-1$ ). Once the system detects the value 1023, the channel is considered to be saturated.

Refer to M221 Hardware Guide and TMC2 Cartridges Hardware Guide used in the configuration for more details.

### Displaying Analog Input Properties

Follow these steps to display properties of the analog inputs:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>I/O objects</b> → <b>Analog inputs</b> . <b>Result:</b> Analog input properties appear on the screen.

## Analog Inputs Properties

This table describes each property of the analog input:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	True/False	False	Indicates whether the input channel is being referenced in a program.
<b>Address</b>	No	%IWO.i	–	Displays the address of the embedded analog input on the controller, where i represents the channel number. If the controller has n analog input channels, the value of i is given as 0...n-1. For example, %IWO.1 is the analog input at the analog input channel number 1 of the logic controller.
		%IWO.x0y	–	Displays the address of the analog output channel on the cartridge, where x is the cartridge number and y is the channel number.
<b>Symbol</b>	Yes	–	–	The symbol associated with this address. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with this input. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of this symbol throughout the program and/or program comments.
<b>Comment</b>	Yes	–	–	The comment associated with this address. Double-click in the <b>Comment</b> column and type a comment to associate with this address.

## Analog Outputs (%QW)

### Introduction

Analog output word objects are the digital values of the analog signals received from the logic controller using cartridges.

Two 0-10 V analog outputs and two 4-20 mA analog outputs are embedded in the cartridges TMC2AQ2C and TMC2AQ2V respectively.

Refer to TMC2 Cartridges Hardware Guide used in the configuration for more details.

### Displaying Analog Output Properties

Follow these steps to display properties of the analog outputs:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>I/O objects</b> → <b>Analog outputs</b> . <b>Result:</b> Analog output properties appear on the screen.

### Analog Outputs Properties

This table describes each property of the analog output:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	True/False	False	Indicates whether the output channel is being referenced in a program.
<b>Address</b>	No	%QW0.x0y	–	Displays the address of the analog output channel on the cartridge, where x is the cartridge number and y is the channel number.
<b>Symbol</b>	Yes	–	–	The symbol associated with this address. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with this output. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of this symbol throughout the program and/or program comments.
<b>Comment</b>	Yes	–	–	The comment associated with this address. Double-click in the <b>Comment</b> column and type a comment to associate with this address.

---

# Chapter 9

## Network Objects

---

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Input Assembly Objects (%QWE)	170
Output Assembly Objects (%IWE)	172
Input Registers Objects (%QWM)	173
Output Registers Objects (%IWM)	175

## Input Assembly Objects (%QWE)

### Introduction

Input assembly objects are the digital values of EtherNet/IP Input assembly frames received on the logic controller.

### Displaying Input Assembly Properties

Follow these steps to display properties of Input assembly objects:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>Network objects</b> → <b>Input assembly</b> . <b>Result:</b> The properties window appears.

## Input Assembly Properties

This table describes each property of an Input assembly object:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
<b>Address</b>	No	%QWEi	–	The address of the Input assembly, where i is the instance identifier. For the maximum number of instances, refer to Maximum Number of Objects ( <i>see page 38</i> ).
<b>Symbol</b>	Yes	–	–	The symbol associated with this address. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of this symbol throughout the program and/or program comments.
<b>Fallback value</b>	Yes	-32768...32767	0	Specify the value to apply to this object when the logic controller enters the STOPPED or an exception state. <b>NOTE:</b> If <b>Maintain values</b> fallback mode is configured, the object retains its current value when the logic controller enters the STOPPED or an exception state. The value 0 is displayed and cannot be edited. For more details, refer to Fallback Behavior ( <i>see SoMachine Basic, Operating Guide</i> ).
<b>Comment</b>	Yes	–	–	A comment associated with this object. Double-click in the <b>Comment</b> column and type an optional comment to associate with this object.

## Output Assembly Objects (%IWE)

### Introduction

Output assembly objects are the digital values of EtherNet/IP Output assembly frames received on the logic controller.

### Displaying Output Assembly Properties

Follow these steps to display properties of Output assembly objects:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>Network objects</b> → <b>Output assembly</b> . <b>Result:</b> The properties window appears.

### Output Assembly Properties

This table describes each property of an Output assembly object:

Parameter	Editable	Value	Default value	Description
<b>Used</b>	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
<b>Address</b>	No	%IWEi	–	The address of the Output assembly, where <i>i</i> is the instance identifier. For the maximum number of instances, refer to Maximum Number of Objects ( <i>see page 38</i> ).
<b>Symbol</b>	Yes	–	–	The symbol associated with this address. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of this symbol throughout the program and/or program comments.
<b>Comment</b>	Yes	–	–	A comment associated with this object. Double-click in the <b>Comment</b> column and type an optional comment to associate with this object.

---

## Input Registers Objects (%QWM)

### Introduction

Input registers objects are the digital values of Modbus mapping table Input registers received on the logic controller.

### Displaying Input Registers Properties

Follow these steps to display properties of Input registers objects:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>Network objects</b> → <b>Input registers</b> . <b>Result:</b> The properties window appears.

## Input Registers Properties

This table describes each property of an Input registers object:

Parameter	Editable	Value	Default value	Description
<b>Used</b>	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
<b>Address</b>	No	%QWMi	–	The address of the Input registers object, where i is the instance identifier. For the maximum number of instances, refer to Maximum Number of Objects ( <a href="#">see page 38</a> ).
<b>Symbol</b>	Yes	–	–	The symbol associated with this address. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of this symbol throughout the program and/or program comments.
<b>Fallback value</b>	Yes	-32768...32767	0	Specify the value to apply to this object when the logic controller enters the STOPPED or an exception state.  <b>NOTE:</b> If <b>Maintain values</b> fallback mode is configured, the object retains its current value when the logic controller enters the STOPPED or an exception state. The value 0 is displayed and cannot be edited. For more details, refer to Fallback Behavior ( <a href="#">see SoMachine Basic, Operating Guide</a> ).
<b>Comment</b>	Yes	–	–	A comment associated with this object. Double-click in the <b>Comment</b> column and type an optional comment to associate with this object.

## Output Registers Objects (%IWM)

### Introduction

Output registers objects are the digital values of Modbus mapping table output registers received on the logic controller.

### Displaying Output Registers Properties

Follow these steps to display properties of Output registers objects:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>Network objects</b> → <b>Output registers</b> . <b>Result:</b> The properties window appears.

### Output Registers Properties

This table describes each property of an Output registers object:

Parameter	Editable	Value	Default value	Description
<b>Used</b>	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
<b>Address</b>	No	%IWMi	–	The address of the Output registers object, where i is the instance identifier. For the maximum number of instances, refer to Maximum Number of Objects ( <a href="#">see page 38</a> ).
<b>Symbol</b>	Yes	–	–	The symbol associated with this address. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of this symbol throughout the program and/or program comments.
<b>Comment</b>	Yes	–	–	A comment associated with this object. Double-click in the <b>Comment</b> column and type an optional comment to associate with this object.



---

# Chapter 10

## Function Blocks

---

### Advanced Expert Input/Output Functions

#### Introduction

The M221 logic controller provides:

- Four fast inputs (%I0.0, %I0.1, %I0.6 and %I0.7)
- Two fast outputs on controller references that contain transistor outputs (%Q0.0 and %Q0.1)

**NOTE:** No fast outputs functions are supported on controller references that contain relay outputs.

The M221 logic controller supports the following expert I/O functions (depending on the reference):

Functions		Description
<b>Counters</b> (Advanced input functions)	Fast Counter	The <b>FC</b> function can execute fast counts of pulses from sensors, switches, and so on.
	High Speed Counter	The <b>HSC</b> function can execute fast counts of pulses from sensors, switches, and so on, that are connected to the fast inputs.
<b>Pulse Generators</b> (Advanced output functions)	Pulse	The <b>PLS</b> function generates a square wave pulse signal on dedicated output channels.
	Pulse Width Modulation	The <b>PWM</b> function generates a modulated wave signal on dedicated output channels with a variable duty cycle.
	Pulse Train Output	The <b>PTO</b> function generates a pulse train output to control a linear single-axis stepper or servo drive in open loop mode.

These functions and their function blocks are described in the Modicon M221 Logic Controller Advanced Functions Library Guide.

**NOTE:**

- When an input is used as Run/Stop, it cannot be used by an expert function.
- When an output is used as Alarm, it cannot be used by an expert function.



---

# Chapter 11

## System Objects

---

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
System Bits (%S)	180
System Words (%SW)	188

## System Bits (%S)

### Introduction

This section provides information about the function of system bits.

### Displaying System Bits Properties

Follow these steps to display properties of the system bits:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>System objects</b> → <b>System Bits</b> . <b>Result:</b> System bit properties appear on the screen.

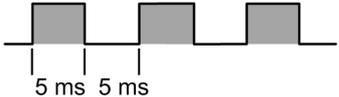
### System Bits Properties

This table describes each property of the system bit:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	TRUE/FALSE	FALSE	Indicates whether the system bit is being referenced in a program.
<b>Address</b>	No	%Si	–	Displays the system bit address, where <i>i</i> is the bit number that represents the sequential position of the system bit in the memory. If the controller has maximum <i>n</i> system bits, the value of <i>i</i> is given as 0... <i>n</i> -1. For example, %S4 is system bit 4.
<b>Symbol</b>	Yes	–	–	The symbol associated with the system bit. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with the system bit. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of the symbol throughout the program and/or program comments.
<b>Comment</b>	Yes	–	–	A comment associated with the system bit. Double-click in the <b>Comment</b> column and type an optional comment to associate with the system bit.

## System Bits Description

This table presents the description of the system bits and how they are controlled:

System Bit	Function	Description	Init State	Control
%S0	Cold Start	Normally set to 0, it is set to 1 by: <ul style="list-style-type: none"> <li>• A power return with loss of data (battery malfunction),</li> <li>• The program or an animation table.</li> </ul> This bit is set to 1 during the first complete scan. It is reset to 0 by the system before the next scan.	0	S or U→S, SIM
%S1	Warm Start Only Read operation is available	Normally set to 0. It is set to 1 by a power return with data backup. It is reset to 0 by the system at the end of the complete scan.	0	S
%S4 %S5 %S6 %S7	Time base: 10 ms Time base: 100 ms Time base: 1 s Time base: 1 min	The rate of status changes is measured by an internal clock. They are not synchronized with the controller scan. Example: %S4 	–	S, SIM (except %S4)
%S10	I/O communication status	Normally set to 1 (TRUE on control panel). This bit can be set to 0 (FALSE on control panel) by the system when an I/O communication interruption is detected. When %S10=0, the <b>ERR</b> LED flashes.	1	S
%S11	Watchdog overflow	Normally set to 0. This bit can be set to 1 by the system when the program execution time (scan time) exceeds the maximum scan time (application watchdog). Watchdog overflow causes the controller state to change to HALTED. %S11 is also set to 1 by the system if the processing load is greater than 80% of the processing capacity (refer to %SW75 (see page 188)). If the processor load is greater than 80% on any two consecutive measurements, the controller goes to HALTED state. Otherwise, %S11 is reset.	0	S
%S12	Logic controller in RUNNING state	This bit indicates that the controller is RUNNING. The system sets the bit to: <ul style="list-style-type: none"> <li>• 1 when the controller state is RUNNING,</li> <li>• 0 for STOPPED, BOOTING, or any other state.</li> </ul>	0	S, SIM
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>U→S</b> Set to 1 by the user, reset to 0 by the system <b>S→U</b> Set to 1 by the system, reset to 0 by the user <b>SIM</b> Applied in the Simulator				

System Bit	Function	Description	Init State	Control
%S13	First cycle in <code>RUNNING</code> state	Normally set to 0. Set to 1 by the system during the first scan after the controller state has been changed to <code>RUNNING</code> .	0	S, SIM
%S14	I/O force activated	Normally set to 0. Set to 1 by the system if at least one input or output is being forced.	0	S, SIM
%S15	Input forced	Normally set to 0. Set to 1 by the system if at least one input is being forced.	0	S, SIM
%S16	Output forced	Normally set to 0. Set to 1 by the system if at least one output is being forced.	0	S, SIM
%S17	Last ejected bit	Normally set to 0. It is set by the system according to the value of the last ejected bit. It indicates the value of the last ejected bit.	0	S→U, SIM
%S18	Arithmetic overflow or error	Normally set to 0. It is set to 1 in the case of an overflow when a 16-bits operation is performed, that is: <ul style="list-style-type: none"> <li>● A result greater than + 32767 or less than - 32768, in single length,</li> <li>● A result greater than + 2147483647 or less than - 2147483648, in double length,</li> <li>● A result greater than + 3.402824E+38 or less than - 3.402824E+38, in floating point,</li> <li>● Division by 0,</li> <li>● The square root of a negative number,</li> <li>● BTI or ITB conversion not significant: BCD value out of limits.</li> </ul> It must be tested by the program after each operation where there is a risk of an overflow; then reset to 0 by the program if an overflow occurs.	0	S→U, SIM
%S19	Scan period overrun (periodic scan)	Normally set to 0, this bit is set to 1 by the system in the event of a scan period overrun (scan time greater than the period defined by the program at configuration or programmed in %SW0). This bit is reset to 0 by the program.	0	S→U
%S20	Index overflow	Normally set to 0, it is set to 1 when the address of the indexed object becomes less than 0 or more than the maximum size of an object. It must be tested by the program, after each operation where there is a risk of overflow; then reset to 0 if an overflow occurs.	0	S→U, SIM
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>U→S</b> Set to 1 by the user, reset to 0 by the system <b>S→U</b> Set to 1 by the system, reset to 0 by the user <b>SIM</b> Applied in the Simulator				

System Bit	Function	Description	Init State	Control
%S21	GRAF CET initialization	<p>Normally set to 0, it is set to 1 by:</p> <ul style="list-style-type: none"> <li>• A cold restart, %S0 = 1,</li> <li>• The program, in the preprocessing program part only, using a Set Instruction (S %S21) or a set coil -(S)- %S21,</li> <li>• The terminal.</li> </ul> <p>At state 1, it causes GRAFCET initialization. Active steps are deactivated and initial steps are activated. It is reset to 0 by the system after GRAFCET initialization.</p>	0	U→S, SIM
%S22	GRAF CET reset	<p>Normally set to 0, it can only be set to 1 by the program in pre-processing.</p> <p>At state 1, it causes the active steps of the entire GRAFCET to be deactivated. It is reset to 0 by the system at the start of the execution of the sequential processing.</p>	0	U→S, SIM
%S23	Preset and freeze GRAFCET	<p>Normally set to 0, it can only be set to 1 by the program in the pre-processing program module.</p> <p>Set to 1, it validates the pre-positioning of GRAFCET. Maintaining this bit at 1 freezes the GRAFCET (freezes the chart). It is reset to 0 by the system at the start of the execution of the sequential processing to ensure that the GRAFCET chart moves on from the frozen situation.</p>	0	U→S, SIM
%S33	Read or Write selection for Ethernet server configuration read/change	<p>Normally set to 0.</p> <ul style="list-style-type: none"> <li>• Set to 0, the %SW33 to %SW38 contains the Ethernet parameters in use (IP declared or IP assigned by BOOTP or automatic IP self assigned). These parameters are those configured in the application or those of the post configuration in SD card (in this case, %SW98 or %SW99 or %SW100 is different from 0).</li> <li>• Set to 1 (if there is no post configuration in use), then the new configuration is given by %SW33 to %SW38.</li> </ul> <p>This bit can be set to its initial state 0 by the program and the system (on cold restart). Then, the Ethernet is reset to apply the application configuration whatever the current configuration is. This bit cannot be set to 1 if a post configuration is in use.</p>	0	U→S
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>U→S</b> Set to 1 by the user, reset to 0 by the system  <b>S→U</b> Set to 1 by the system, reset to 0 by the user  <b>SIM</b> Applied in the Simulator</p>				

System Bit	Function	Description	Init State	Control
%S34	Ethernet Autonegotiation	Set to 0 to allow the autonegotiation of the speed and half or full duplex mode. Set to 1 to force some specific configuration set in %S35 and %S36.  <b>NOTE:</b> A modification in the state of %S34, %S35, or %S36 provokes a reinitialization of the Ethernet channel, so, after the modification, and during a few minutes, the Ethernet channel is not available.	0	U
%S35	Ethernet half/full duplex mode	In case of the %S34 = 0 (autonegotiation) this bit will be set by the system, and it will be read only for the user. But is the %S34 = 1, the mode will be forced based on the value of this bit set by the user: <ul style="list-style-type: none"> <li>● Set to 0 if Half Duplex,</li> <li>● Set to 1 if Full Duplex.</li> </ul> <b>NOTE:</b> A modification in the state of %S34, %S35, or %S36 provokes a reinitialization of the Ethernet channel, so, after the modification, and during a few minutes, the Ethernet channel is not available.	–	U or S
%S36	Ethernet speed	In case of the %S34 = 0 (autonegotiation) this bit will be set by the system, and it will be read only for the user. But is the %S34 = 1, the mode will be forced based on the value of this bit set by the user: <ul style="list-style-type: none"> <li>● Set to 0 if 10 Mbps,</li> <li>● Set to 1 if 100 Mbps.</li> </ul> <b>NOTE:</b> A modification in the state of %S34, %S35, or %S36 provokes a reinitialization of the Ethernet channel, so, after the modification, and during a few minutes, the Ethernet channel is not available.	–	U or S
%S38	Permission for events to be placed in the events queue	Normally set to 1. <ul style="list-style-type: none"> <li>● Set to 0, events cannot be placed in the events queue.</li> <li>● Set to 1, events are placed in the events queue as soon as they are detected,</li> </ul> This bit can be set to its initial state 1 by the program and the system (on cold restart).	1	U→S
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>U→S</b> Set to 1 by the user, reset to 0 by the system <b>S→U</b> Set to 1 by the system, reset to 0 by the user <b>SIM</b> Applied in the Simulator				

System Bit	Function	Description	Init State	Control
%S39	Saturation of the events queue	<p>Normally set to 0.</p> <ul style="list-style-type: none"> <li>Set to 0, all events are reported.</li> <li>Set to 1, at least one event is lost.</li> </ul> <p>This bit can be set to 0 by the program and the system (on cold restart).</p>	0	U→S
%S49	Output rearming ( <i>see page 57</i> )	<p>Normally set to 0, this bit can be set to 1 or 0 by the program.</p> <ul style="list-style-type: none"> <li>Set to 0, the automatic re-arming of outputs following a short circuit is disabled.</li> <li>Set to 1, the automatic re-arming of outputs following a short circuit is enabled.</li> </ul> <p><b>NOTE:</b> The bit is reset to 0 on a cold start; otherwise, the bit value is retained.</p> <p>The system bit %S10 can be used to detect within your program that an output error has occurred. You can then use the system word %SW139 to determine programmatically in which cluster of outputs a short circuit or overload has occurred.</p> <p><b>NOTE:</b> %S10 and %SW139 are reset to their initial state when %S49 is set to 1.</p>	0	U→S
%S50	Updating the date and time using words %SW49 to %SW53	<p>Normally set to 0, this bit can be set to 1 or 0 by the program.</p> <ul style="list-style-type: none"> <li>Set to 0, the date and time can be read.</li> <li>Set to 1, the date and time can be updated.</li> </ul> <p>The internal RTC controller is updated on a falling edge of %S50.</p>	0	U→S
%S51	Time-of-day clock status	<p>Normally set to 0, this bit can be set to 1 or 0 by the program.</p> <ul style="list-style-type: none"> <li>Set to 0, the date and time are consistent.</li> <li>Set to 1, the date and time must be initialized by the program.</li> </ul> <p>When this bit is set to 1, the time of day clock data is not valid. The date and time may never have been configured, the battery may be low, or the controller correction constant may be invalid (never configured, difference between the corrected clock value and the saved value, or value out of range).</p> <p>State 1 transitioning to state 0 forces a Write of the correction constant to the RTC.</p>	0	U→S, SIM
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>U→S</b> Set to 1 by the user, reset to 0 by the system  <b>S→U</b> Set to 1 by the system, reset to 0 by the user  <b>SIM</b> Applied in the Simulator</p>				

System Bit	Function	Description	Init State	Control
%S52	RTC error detected	This bit, managed by the system, indicates that the RTC correction has not been entered. Therefore, the date and time are assumed to be incorrect. <ul style="list-style-type: none"> <li>At state 0, the RTC correction has been entered and the date and time are assumed to be correct,</li> <li>At state 1, the RTC correction has not been entered and the date and time are assumed to be incorrect, and must be initialized.</li> </ul>	0	S, SIM
%S59	Updating the date and time using word %SW59	Normally set to 0, this bit can be set to 1 or 0 by the program. <ul style="list-style-type: none"> <li>Set to 0, the system word %SW59 is not managed,</li> <li>Set to 1, the date and time are incremented or decremented according to the rising edges on the control bits set in %SW59.</li> </ul>	0	U
%S75	Battery status	This system bit is set by the system and can be read by the user. It indicates the battery status: <ul style="list-style-type: none"> <li>Set to 0, the external battery is operating normally.</li> <li>Set to 1, external battery power is low, or no external battery is detected.</li> </ul>	0	S
%S91	Erase persistent variables in non-volatile memory	Set to 1 to erase the persistent variables stored in non-volatile memory.	–	U→S
%S92	%MW variables saved in non-volatile memory	Set to 1 if there are memory word (%MW) variables stored in non-volatile memory. Set to 0 if the data block is invalid or the write operation is in progress.	–	S
%S93	Backup %MW in non-volatile memory	Set this bit to 1 to store the %MW variables in the non-volatile memory (up to 1000).	–	U→S
%S94	Restore %MW	Set this bit to 1 to restore the data saved in non-volatile memory.	–	U→S
%S96	Backup program OK	This bit can be read at any time (either by the program or while adjusting), in particular after a cold start or a warm restart. <ul style="list-style-type: none"> <li>Set to 0, the backup program is invalid.</li> <li>Set to 1, the backup program is valid.</li> </ul>	0	S, SIM
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>U→S</b> Set to 1 by the user, reset to 0 by the system <b>S→U</b> Set to 1 by the system, reset to 0 by the user <b>SIM</b> Applied in the Simulator				

System Bit	Function	Description	Init State	Control
%S101	Changing a port address (Modbus protocol)	Used to change a port address using system words %SW101 (port 1) and %SW102 (port 2). To do this, %S101 must be set to 1. <ul style="list-style-type: none"> <li>Set to 0, the address cannot be changed. The value of %SW101 and %SW102 matches the current port address,</li> <li>Set to 1, the address can be changed by changing the values of %SW101 (port 1) and %SW102 (port 2).</li> </ul> <p><b>NOTE:</b> After a cold start (%S0=1) or a warm start (%s1=1) all dynamic values are lost and the initial port address values are restored.</p> <p><b>NOTE:</b> %SW101 cannot be set to 1 if a post configuration file is defined on port 1 or port 2.</p>	0	U
%S103 %S104	Using the ASCII protocol	Enables the use of the ASCII protocol on Comm 1 (%S103) or Comm 2 (%S104). The ASCII protocol is configured using system words %SW103 and %SW105 for Comm 1, and %SW104 and %SW106 for Comm 2. <ul style="list-style-type: none"> <li>Set to 0, the protocol used is the one configured in SoMachine Basic,</li> <li>Set to 1, the ASCII protocol is used on Comm 1 (%S103) or Comm 2 (%S104). In this case, the system words %SW103, %SW105, and %sw121 must be previously configured for COM 1, and %SW104, %SW106, and %SW122 for COM 2. Each change of those %SW will be taken into account after a rising edge to %S103 or %S104.</li> </ul> <p><b>NOTE:</b> A rising or falling edge on %S103 or %S104 cancels an exchange in progress (EXCH instruction).</p> <p><b>NOTE:</b> Setting %S103 or %S104 to 0 reconfigures the serial line with the SoMachine Basic parameters.</p>	0	U
%S105	Modem initialization command	Set to 1 to send the initialization command to the modem. Reset to 0 by the system. See also %SW167 ( <a href="#">see page 188</a> ).	0	U/S
%S119	Local I/O error detected	Normally set to 1. This bit can be set to 0 when an I/O communication interruption is detected on the logic controller. %SW118 determines the nature of the communication interruption. Resets to 1 when the communication interruption disappears.	1	S
%S122	Automatically switch to Alarm page	Set to 1, the Remote Graphic Display automatically switches to the <b>Alarm</b> page when a rising edge is detected on an alarm bit.	0	U
%S123	Display red backlight on alarm	Set to 1, the backlight on the Remote Graphic Display is red when an alarm is active.	0	U
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>U→S</b> Set to 1 by the user, reset to 0 by the system  <b>S→U</b> Set to 1 by the system, reset to 0 by the user  <b>SIM</b> Applied in the Simulator</p>				

## System Words (%SW)

### Introduction

This section provides information about the function of system words.

### Displaying System Words Properties

Follow these steps to display properties of the system words:

Step	Action
1	Select the <b>Tools</b> tab in the left-hand area of the <b>Programming</b> window.
2	Click <b>System objects</b> → <b>System Words</b> . <b>Result:</b> System word properties appear on the screen.

### System Words Properties

This table describes each property of the system word:

Parameter	Editable	Value	Default Value	Description
<b>Used</b>	No	TRUE/FALSE	FALSE	Indicates whether the system word is being referenced in a program.
<b>Address</b>	No	%SWi	–	Displays the system word address, where i is the word number that represents the sequential position of the system word in the memory. If the controller has maximum n system words, the value of i is given as 0...n-1. For example, %SW50 is system word 50.
<b>Symbol</b>	Yes	–	–	The symbol associated with the system word. Double-click in the <b>Symbol</b> column and type the name of the symbol to associate with the system word. If a symbol already exists, you can right-click in the <b>Symbol</b> column and choose <b>Search and Replace</b> to find and replace occurrences of the symbol throughout the program and/or program comments.
<b>Comment</b>	Yes	–	–	A comment associated with the system word. Double-click in the <b>Comment</b> column and type an optional comment to associate with the system word.

## System Words Description

This table presents the description of the system words and how they are controlled:

System Words	Function	Description	Control
%SW0	Controller scan period (master task set to periodic scan mode)	Modifies controller scan period defined at configuration through the program in an animation table.	U, SIM
%SW1	Periodic task period	Modifies the cycle time [2...255 ms] of the periodic task, without losing the <b>Period</b> value specified in the periodic task properties window. Allows you to recover the <b>Period</b> value saved in the periodic task properties window: <ul style="list-style-type: none"> <li>● in case of a cold start, or</li> <li>● if the value you write in %SW1 is outside [2...255] range.</li> </ul> The %SW1 value can be modified in the program at each end of a cycle, in the program or in an animation table without having to stop the program. Cycle times can be correctly observed while the program is running.	U, SIM
%SW6	Controller state %MW60012	Controller state: 0 = EMPTY 2 = STOPPED 3 = RUNNING 4 = HALTED 5 = POWERLESS	S, SIM
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control
%SW7	Controller status	<ul style="list-style-type: none"> <li>● Bit [0]: Backup/restore in progress: <ul style="list-style-type: none"> <li>○ Set to 1 if backup/restore of the program is in progress,</li> <li>○ Set to 0 if backup/restore of the program is complete or disabled.</li> </ul> </li> <li>● Bit [1]: Configuration of the controller is OK: <ul style="list-style-type: none"> <li>○ Set to 1 if configuration ok.</li> </ul> </li> <li>● Bit [2]: SD card status bits: <ul style="list-style-type: none"> <li>○ Set to 1 if SD card is present.</li> </ul> </li> <li>● Bit [3]: SD card status bits: <ul style="list-style-type: none"> <li>○ Set to 1 if SD card is being accessed.</li> </ul> </li> <li>● Bit [4]: Application memory status: <ul style="list-style-type: none"> <li>○ Set to 1 if application in RAM memory is different from that in non-volatile memory.</li> </ul> </li> <li>● Bit [6]: Not used (status 0)</li> <li>● Bit [7]: Controller reserved: <ul style="list-style-type: none"> <li>○ Set to 1 when the controller is connected to SoMachine Basic.</li> </ul> </li> <li>● Bit [8]: Application in Write mode: <ul style="list-style-type: none"> <li>○ Set to 1 if application is protected. In this case, the clone operation does not replicate the application (refer to Clone Management (<a href="#">see page 148</a>)).</li> </ul> </li> <li>● Bit [9]: Not used (status 0)</li> <li>● Bit [10]: Second serial port installed as cartridge (compact only): <ul style="list-style-type: none"> <li>○ 0 = No serial cartridge</li> <li>○ 1 = Serial cartridge installed</li> </ul> </li> <li>● Bit [11]: Second serial port type: <ul style="list-style-type: none"> <li>○ Set to 1 = EIA RS-485</li> </ul> </li> <li>● Bit [12]: Validity of the application in internal memory: <ul style="list-style-type: none"> <li>○ Set to 1 if the application is valid.</li> </ul> </li> <li>● Bit [14]: Validity of the application in RAM memory: <ul style="list-style-type: none"> <li>○ Set to 1 if the application is valid.</li> </ul> </li> <li>● Bit [15]: Ready for execution: <ul style="list-style-type: none"> <li>○ Set to 1 if ready for execution.</li> </ul> </li> </ul>	S, SIM
%SW11	Software watchdog value	Contains the maximum value of the watchdog. The value (10...500 ms) is defined by the configuration.	U, SIM
%SW13	Boot loader version xx.yy	<p>For example, if %SW13=000E hex:</p> <ul style="list-style-type: none"> <li>● 8 MSB=00 in hexadecimal, then xx=0 in decimal</li> <li>● 8 LSB=0E in hexadecimal, then yy=14 in decimal</li> </ul> <p>As a result, boot loader version is 0.14, displayed as 14 decimal.</p>	S, SIM
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>SIM</b> Applied in the simulator</p>			

System Words	Function	Description	Control
%SW14	Commercial version, xx.yy	For example, if %SW14=0232 hex: <ul style="list-style-type: none"> <li>● 8 MSB=02 in hexadecimal, then xx=2 in decimal</li> <li>● 8 LSB=32 in hexadecimal, then yy=50 in decimal</li> </ul> As a result, commercial version is 2.50, displayed as 250 decimal.	S, SIM
%SW15– %SW16	Firmware version aa.bb.cc.dd	For example, if: %SW15=0003 hex: <ul style="list-style-type: none"> <li>● 8 MSB=00 in hexadecimal, then aa=00 in decimal</li> <li>● 8 LSB=03 in hexadecimal, then bb=03 in decimal</li> </ul> %SW16=0B16 hex: <ul style="list-style-type: none"> <li>● 8 MSB=0B in hexadecimal, then cc=11 in decimal</li> <li>● 8 LSB=16 in hexadecimal, then dd=22 in decimal</li> </ul> As a result, firmware version is 0.3.11.22 displayed as 00031122 decimal.	S, SIM
%SW17	Default status for floating operation	When an error is detected in a floating arithmetic operation, bit %S18 is set to 1 and the default status of %SW17 is updated according to the following coding: <ul style="list-style-type: none"> <li>● Bit[0]: Invalid operation, result is not a number (NaN)</li> <li>● Bit[1]: Reserved</li> <li>● Bit[2]: Division by 0, result is invalid (-Infinity or +Infinity)</li> <li>● Bit[3]: Result greater in absolute value than +3.402824e+38, result is invalid (-Infinity or +Infinity).</li> </ul>	S and U, SIM
%SW18– %SW19	100 ms absolute timer counter	This counter works using 2 words: <ul style="list-style-type: none"> <li>● %SW18 represents the least significant word,</li> <li>● %SW19 represents the most significant word.</li> </ul> The double word (%SW18-%SW19) increases from 0 to 2 <sup>31</sup> each 100 ms as a counter modulo 2 <sup>31</sup> . This double word is also reset during the INIT phase and on a reset of %S0.	S and U, SIM
%SW30	Last scan time (master task)	Indicates the execution time of the last controller scan cycle (in ms).  <b>NOTE:</b> This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a master task scan cycle. If the scan time is 2.250 ms, the %SW30 is 2 and the %SW70 is 250.	S
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control
%SW31	Max. scan time (master task)	Indicates the execution time of the longest controller scan cycle since the last cold start (in ms). This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a scan cycle. If the maximum scan time is 2.250 ms, the %SW31 will be 2 and the %SW71 will be 250.  <b>NOTE:</b> To detect a pulse signal when the latching input option is selected, the pulse width ( $T_{ON}$ ) and the period (P) must meet the following 2 requirements: <ul style="list-style-type: none"> <li>• <math>T_{ON} \geq 1</math> ms</li> <li>• The input signal period (P) follows the Nyquist-Shannon sampling rule stating that the input signal period (P) is at least twice the maximum program scan time (%SW31): <math>P \geq 2 \times \%SW31</math>.</li> </ul>	S
%SW32	Min. scan time (master task)	Indicates the execution time of shortest controller scan cycle since the last cold start (in ms).  <b>NOTE:</b> This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a scan cycle. If the minimum scan time is 2.250 ms, the %SW32 is 2 and the %SW72 is 250.	S
%SW33 %SW34 %SW35 %SW36 %SW37 %SW38	IP address for Ethernet server configuration read/write	The IP settings can be modified. The read or write selection is done using the system bit %S33. The system words %SW33 . . . %SW38 contains the Ethernet parameters: <ul style="list-style-type: none"> <li>• IP address: %SW33 and %SW34 For IP address AA.BB.CC.DD: %SW33 = CC.DD and %SW34 = AA.BB</li> <li>• Subnetwork mask: %SW35 and %SW36 For subnetwork mask AA.BB.CC.DD: %SW35 = CC.DD and %SW36 = AA.BB</li> <li>• Gateway address: %SW37 and %SW38 For gateway address AA.BB.CC.DD: %SW37 = CC.DD and %SW38 = AA.BB</li> </ul>	U
%SW39	Periodic average time	Indicates the average execution time in $\mu$ s of the periodic task (last 5 times).	–
%SW40	Event 0 average time	Indicates the average execution time in $\mu$ s of the event task associated with the input %IO.2 (last 5 times).	–
%SW41	Event 1 average time	Indicates the average execution time in $\mu$ s of the event task associated with the input %IO.3 (last 5 times).	–
%SW42	Event 2 average time	Indicates the average execution time in $\mu$ s of the event task associated with the input %IO.4 (last 5 times).	–
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control
%SW43	Event 3 average time	Indicates the average execution time in $\mu$ s of the event task associated with the input %I0.5 (last 5 times).	–
%SW44	Event 4 average time	Indicates the average execution time in $\mu$ s of the event task associated with the Threshold 0 of HSC0 or HSC2 (last 5 times).	–
%SW45	Event 5 average time	Indicates the average execution time in $\mu$ s of the event task associated with the Threshold 1 of HSC0 or HSC2 (last 5 times).	–
%SW46	Event 6 average time	Indicates the average execution time in $\mu$ s of the event task associated with the Threshold 0 of HSC1 or HSC3 (last 5 times).	–
%SW47	Event 7 average time	Indicates the average execution time in $\mu$ s of the event task associated with the Threshold 1 of HSC1 or HSC3 (last 5 times).	–
%SW48	Number of events	Indicates how many events have been executed since the last cold start. (Counts all events except cyclic events.) <b>NOTE:</b> Set to 0 (after application loading and cold start), increments on each event execution.	S, SIM
%SW49 %SW50 %SW51 %SW52 %SW53	Real-Time Clock (RTC)	RTC functions: words containing current date and time values (in BCD): %SW49 xN Day of the week (N=1 for Monday) <b>NOTE:</b> %SW49 is read only (S). %SW50 00SS Seconds %SW51 HHMM: hour and minute %SW52 MMDD: month and day %SW53 CCYY: century and year These words are controlled by the system when bit %S50 is at 0. These words can be written by the program or by the terminal when bit %S50 is set to 1. On a falling edge of %S50 the internal RTC controller is updated from the values written in these words.	S and U, SIM
%SW54 %SW55 %SW56 %SW57	Date and time of the last stop	System words containing the date and time of the last power outage or controller stop (in BCD): %SW54 SS Seconds %SW55 HHMM: hour and minute %SW56 MMDD: month and day %SW57 CCYY: century and year	S, SIM
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control			
%SW58	Code of last stop	Displays code giving cause of last stop:	S, SIM			
		1 =		Run/Stop input edge		
		2 =		Stop at detected software error (controller scan overshoot)		
		3 =		Stop command (SoMachine Basic online button or Remote Graphic Display)		
		4 =		Power outage		
		5 =		Stop at detected hardware error		
		6 =		Init in cold start		
		7 =		Start in stop		
		8 =		Low battery		
		9 =		Controller is not OK to run		
%SW59	Adjust current date	Adjusts the current date. Contains 2 sets of 8 bits to adjust current date. The operation is always performed on rising edge of the bit. This word is enabled by bit %S59.	U			
		<b>Increment</b>		<b>Decrement</b>	<b>Parameter</b>	
		bit 0		bit 8	Day of week	Not used
		bit 1		bit 9	Seconds	
		bit 2		bit 10	Minutes	
		bit 3		bit 11	Hours	
		bit 4		bit 12	Days	
		bit 5		bit 13	Month	
		bit 6		bit 14	Years	
		bit 7		bit 15	Centuries	Not used
%SW62	Ethernet error detection	Indicates the error code: 0 - No error detected 1 - Duplicate IP: the M221 Logic Controller is configured with its default IP address (generated from the MAC address) 2 - DHCP in progress 3 - BOOTP in progress 4 - Invalid parameters: port is disabled 5 - Fixed IP address initialization in progress 6 - Ethernet link down	S			
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator						

System Words	Function	Description	Control
%SW63	EXCH1 block error code	EXCH1 error code: 0 - operation was successful 1 - number of bytes to be transmitted exceeds the limit (> 255) 2 - insufficient transmission table 3 - insufficient word table 4 - receive table overflowed 5 - time-out elapsed 6 - transmission 7 - incorrect command within table 8 - selected port not configured/available 9 - reception error: This error code reflects an incorrect or corrupted reception frame. It can be caused due to an incorrect configuration in the physical parameters (for example, parity, data bits, baudrate, and so on) or an unreliable physical connection causing signal degradation. 10 - cannot use %KW if receiving 11 - transmission offset larger than transmission table 12 - reception offset larger than reception table 13 - controller stopped EXCH processing	S
%SW64	EXCH2 block error code	EXCH2 error code: See %SW63.	S
%SW65	EXCH3 block error code	1-4, 6-13: See %SW63. (Note that error code 5 is invalid and replaced by the Ethernet-specific error codes 109 and 122 described below.) The following are Ethernet-specific error codes: 101 - incorrect IP address 102 - no TCP connection 103 - no socket available (all connection channels are busy) 104 - network is down 105 - network cannot be reached 106 - network dropped connection on reset 107 - connection aborted by peer device 108 - connection reset by peer device 109 - connection time-out elapsed 110 - rejection on connection attempt 111 - host is down 120 - incorrect index (remote device is not indexed in configuration table) 121 - system error (MAC, chip, duplicate IP) 122 - receiving process timed-out after data was sent 123 - Ethernet initialization in progress	S
%SW67	Function and type of controller	Contains the logic controller code ID. For more information, refer to the M221 Logic Controller Code ID table ( <a href="#">see page 209</a> ).	S, SIM
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control
%SW70	Scan time microseconds resolution	Indicates the execution time of the last controller scan cycle (in $\mu$ s). <b>NOTE:</b> This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a master task scan cycle. If the scan time is 2.250 ms, the %SW30 will be 2 and the %SW70 will be 250.	–
%SW71	Max. scan time microseconds resolution	Indicates the execution time of the longest controller scan cycle since the last cold start (in ms). <b>NOTE:</b> This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a scan cycle. If the scan time is 2.250 ms, the %SW31 will be 2 and the %SW71 will be 250.	–
%SW72	Min. scan time microseconds resolution	Indicates the execution time of the shortest controller scan cycle since the last cold start (in ms). <b>NOTE:</b> This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a scan cycle. If the scan time is 2.250 ms, the %SW32 will be 2 and the %SW72 will be 250.	–
%SW75	Load of processor	Indicates the percentage of processing load. Processing load is defined as the percentage of the total available processing time that is used to process your program tasks (this value is an average and it is calculated every second). In case of processing load higher than 80% for two consecutive periods of time, the controller goes to HALTED state.	S
%SW76 to %SW79	Down counters 1-4	These 4 words serve as 1 ms timers. They are decremented individually by the system every ms if they have a positive value. This gives 4 down counters down counting in ms which is equal to an operating range of 1 ms to 32767 ms. Setting bit 15 to 1 can stop decrementation.	S and U, SIM
%SW94 %SW95	Application's signature %MW60028– %MW60034	In case of an application change, in terms of configuration or programming data, the signature (sum of all checksums) changes so. If %SW94 = 91F3 in hexadecimal, the application's signature is 91F3 in hexadecimal.	S, SIM
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control
%SW96	Diagnostics for save/restore function of program and %MW	<ul style="list-style-type: none"> <li>● Bit [1]: This bit is set by the firmware to indicate when the save is complete: <ul style="list-style-type: none"> <li>○ Set to 1 if the backup is complete.</li> <li>○ Set to 0 if a new backup is requested.</li> </ul> </li> <li>● Bit [2]: Back up error detected, refer to bits 8, 9, 10, 12 and 14 for further information: <ul style="list-style-type: none"> <li>○ Set to 1 if an error is detected.</li> <li>○ Set to 0 if a new backup is requested.</li> </ul> </li> <li>● Bit [6]: Set to 1 if the controller contains a valid application in RAM memory.</li> <li>● Bit [10]: Difference detected between RAM memory and non-volatile memory. <ul style="list-style-type: none"> <li>○ Set to 1 if there is a difference.</li> </ul> </li> <li>● Bit [12]: Indicates if a restore error has occurred: <ul style="list-style-type: none"> <li>○ Set to 1 if an error is detected.</li> </ul> </li> <li>● Bit [14]: Indicates if a non-volatile memory write error has occurred: <ul style="list-style-type: none"> <li>○ Set to 1 if an error is detected.</li> </ul> </li> </ul>	S, SIM
%SW98	Post configuration status (Serial Line 1)	<p>The bits are set to 1 when the post configuration was applied for the parameter:</p> <ul style="list-style-type: none"> <li>● Bit[0]: Hardware option (RS485 or RS232)</li> <li>● Bit[1]: Baudrate</li> <li>● Bit[2]: Parity</li> <li>● Bit[3]: Data size</li> <li>● Bit[4]: Number of stop bits</li> <li>● Bit[5]: Modbus address</li> <li>● Bit[6]: Polarization (if available in the port)</li> </ul>	S
%SW99	Post configuration status (Serial Line 2)	<p>The bits are set to 1 when the post configuration was applied for the parameter:</p> <ul style="list-style-type: none"> <li>● Bit[0]: Hardware option (RS485)</li> <li>● Bit[1]: Baudrate</li> <li>● Bit[2]: Parity</li> <li>● Bit[3]: Data size</li> <li>● Bit[4]: Number of stop bits</li> <li>● Bit[5]: Modbus address</li> <li>● Bit[6]: Polarization (if available in the port)</li> </ul>	S
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>SIM</b> Applied in the simulator</p>			

System Words	Function	Description	Control
%SW100	Post configuration status (Ethernet)	<p>The bits are set to 1 when the post configuration was applied for the parameter:</p> <ul style="list-style-type: none"> <li>● Bit[0]: IP mode (fixed, DHCP, or BOOTP)</li> <li>● Bit[1]: IP address</li> <li>● Bit[2]: Network submask</li> <li>● Bit[3]: Default gateway</li> <li>● Bit[4]: Device name</li> </ul> <p><b>NOTE:</b> The post configuration has priority over the configuration provided by your application. The configuration of your application is not taken into account if the M221 Logic Controller has a post configuration.</p>	S
%SW101 %SW102	Value of the Modbus address port	<p>When bit %S101 is set to 1, you can change the Modbus address of port 1 or port 2. The address of port 1 is %SW101. The address of port 2 is %SW102.</p> <p><b>NOTE:</b> The update is applied immediately after writing a new address to %SW101 or %SW102.</p> <p><b>NOTE:</b> After a cold start (%S0=1) or a warm start (%S1=1) all dynamic values are lost and the initial port address values are restored. %S101 is also updated.</p>	U
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>SIM</b> Applied in the simulator</p>			

System Words	Function	Description	Control																																
%SW103 %SW104	Configuration for use of the ASCII protocol	<p>When bit %S103 (Comm 1) or %S104 (Comm 2) is set to 1, the ASCII protocol is used. System word %SW103 (Comm 1) or %SW104 (Comm 2) must be set according to the elements below:</p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td colspan="8">End of the character string</td> <td>Data bit</td> <td>Stop bit</td> <td>Parity</td> <td>RTS / CTS</td> <td colspan="4">Baud rate</td> </tr> </table> <ul style="list-style-type: none"> <li>● Baud rate: <ul style="list-style-type: none"> <li>○ 000: 1200 baud,</li> <li>○ 001: 2400 baud,</li> <li>○ 010: 4800 baud,</li> <li>○ 011: 9600 baud,</li> <li>○ 100: 19200 baud,</li> <li>○ 101: 38400 baud,</li> <li>○ 110: 57600 baud,</li> <li>○ 111: 115200 baud.</li> </ul> </li> <li>● RTS/CTS: <ul style="list-style-type: none"> <li>○ 0: disabled,</li> <li>○ 1: enabled.</li> </ul> </li> <li>● Parity: <ul style="list-style-type: none"> <li>○ 00: none,</li> <li>○ 10: odd,</li> <li>○ 11: even.</li> </ul> </li> <li>● Stop bit: <ul style="list-style-type: none"> <li>○ 0: 1 stop bit,</li> <li>○ 1: 2 stop bits.</li> </ul> </li> <li>● Data bits: <ul style="list-style-type: none"> <li>○ 0: 7 data bits,</li> <li>○ 1: 8 data bits.</li> </ul> </li> </ul>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	End of the character string								Data bit	Stop bit	Parity	RTS / CTS	Baud rate				S, U
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																				
End of the character string								Data bit	Stop bit	Parity	RTS / CTS	Baud rate																							
%SW105 %SW106	Configuration for use of the ASCII protocol	<p>When bit %S103 (Comm 1) or %S104 (Comm 2) is set to 1, the ASCII protocol is used. System word %SW105 (Comm 1) or %SW106 (Comm 2) must be set according to the elements below:</p> <table border="1" style="width: 100%; text-align: center;"> <tr> <td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td> </tr> <tr> <td colspan="8">Timeout frame in ms</td> <td colspan="8">Timeout response in multiples of 100 ms</td> </tr> </table>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Timeout frame in ms								Timeout response in multiples of 100 ms								S, U
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																				
Timeout frame in ms								Timeout response in multiples of 100 ms																											
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>SIM</b> Applied in the simulator</p>																																			

System Words	Function	Description	Control
%SW107 %SW108 %SW109	MAC address	Indicates the controller MAC address (only references with Ethernet channel). For MAC address AA:BB:CC:DD:EE:FF: <ul style="list-style-type: none"> <li>● %SW107 = AA:BB</li> <li>● %SW108 = CC:DD</li> <li>● %SW109 = EE:FF</li> </ul>	S
%SW114	Enable schedule blocks	Enables or disables operation of schedule blocks by the program: <ul style="list-style-type: none"> <li>● Bit [0]: Enable/disable schedule block number 0 <ul style="list-style-type: none"> <li>○ Set to 0: disabled</li> <li>○ Set to 1: enabled</li> </ul> </li> <li>● ...</li> <li>● Bit [15]: Enable/disable schedule block number 15 <ul style="list-style-type: none"> <li>○ Set to 0: disabled</li> <li>○ Set to 1: enabled</li> </ul> </li> </ul> <p>Initially all schedule blocks are enabled. The default value is FFFF hex.</p>	S and U, SIM
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>SIM</b> Applied in the simulator</p>			

System Words	Function	Description	Control
%SW118	Logic controller status word	<p>Indicates conditions on logic controller. For a controller operating normally, the value of this word is FFFF hex.</p> <ul style="list-style-type: none"> <li>● Bit [9]: <ul style="list-style-type: none"> <li>○ Set to 0: External error detected or communication interruption, for example duplicate IP address</li> <li>○ Set to 1: No error detected.</li> </ul> </li> <li>● Bit [10]: <ul style="list-style-type: none"> <li>○ Set to 0: Invalid internal configuration; contact Schneider Electric customer service.</li> <li>○ Set to 1: No error detected.</li> </ul> </li> <li>● Bit [13]: <ul style="list-style-type: none"> <li>○ Set to 0: Configuration error detected (mandatory modules, as defined by the I/O expansion bus configuration, are absent or otherwise inoperative when the logic controller attempts to start the I/O expansion bus). In this case, the I/O bus does not start.</li> <li>○ Set to 1: No error detected.</li> </ul> </li> <li>● Bit [14]: <ul style="list-style-type: none"> <li>○ Set to 0: One or more modules have ceased communication with the logic controller after the I/O expansion bus is started. This is the case whether an I/O expansion module is defined as mandatory or optional but present at start-up.</li> <li>○ Set to 1: No error detected.</li> </ul> </li> <li>● Bit [15]: <ul style="list-style-type: none"> <li>○ Set to 0: Cartridge error detected (configuration or runtime operation).</li> <li>○ Set to 1: No error detected.</li> </ul> </li> </ul> <p><b>NOTE:</b> The other bits of this word are set to 1 and are reserved.</p>	S, SIM
%SW119	<b>Optional module</b> feature configuration	<p>One bit for each expansion module in the configuration:</p> <ul style="list-style-type: none"> <li>● Bit [0]: Reserved for the logic controller</li> <li>● Bit n: Module n <ul style="list-style-type: none"> <li>○ Set to 1: Module is marked as optional in the configuration.</li> <li>○ Set to 0: Module is not marked as optional in the configuration.</li> </ul> </li> </ul>	S, SIM
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>SIM</b> Applied in the simulator</p>			

System Words	Function	Description	Control
%SW120	Expansion I/O module status	<p>1 bit for each expansion module in the configuration.</p> <p>Bit 0: Reserved for the logic controller</p> <p>When the logic controller attempts to start the I/O bus, bit n:</p> <ul style="list-style-type: none"> <li>● 0 = no error detected</li> <li>● 1 = error detected or module not present. The I/O expansion bus does not start unless the corresponding bit in %SW119 is set to TRUE (indicating the module is marked as optional).</li> </ul> <p>When the I/O bus is started, bit n:</p> <ul style="list-style-type: none"> <li>● 0 = no error detected</li> <li>● 1 = error detected on the I/O expansion module (regardless it is a module marked as optional).</li> </ul>	S, SIM
%SW121 %SW122	Configuration for use of ASCII protocol	<p>When bit %S103 (Comm 1) or %S104 (Comm 2) is set to 1, the ASCII protocol is used. You can change the ASCII frame size of port 1 or port 2. The ASCII frame size of port 1 is %SW121, and that of port 2 is %SW122.</p> <p>The value is used only on EXCH instruction start. Then, if some bytes are already received, you cannot stop the reception until the last byte.</p>	U
<p><b>S</b> Controlled by the system</p> <p><b>U</b> Controlled by the user</p> <p><b>SIM</b> Applied in the simulator</p>			

System Words	Function	Description	Control
%SW128	Cartridge 1 status	Indicates the status code for the cartridge: ● LSB: presents the status of the I/O channel 1 ● MSB: presents the status of the I/O channel 2  General status: ● 0x80: Cartridge is not present and it is not configured in SoMachine Basic. ● 0x81: Module is present, but it is not configured. ● 0x82: Internal communication error with the cartridge. ● 0x83: Internal communication error with the cartridge. ● 0x84: Detected cartridge different from the configuration. ● 0x85: Configured cartridge is not detected.  Input channel operation status: ● 0x00: Normal. ● 0x01: Conversion in progress. ● 0x02: Initialization. ● 0x03: Input operation setting error detected or module without input. ● 0x04: Reserved. ● 0x05: Wiring error detected (High limit range out). ● 0x06: Wiring error detected (Low limit range out). ● 0x07: Non-volatile memory error detected. ● Others: Reserved.  Output channel operation status: ● 0x00: Normal. ● 0x01: Reserved. ● 0x02: Initialization. ● 0x03: Output operation setting error detected or module without output. ● 0x04: Reserved. ● 0x05: Reserved. ● 0x06: Reserved. ● 0x07: Non-volatile memory error detected. ● Others: Reserved.	S, SIM
%SW129	Cartridge 2 status		
%SW130	Event execution time	Indicates the last execution time in $\mu$ s of the event task associated with the input %I0.2.	S
%SW131	Event execution time	Indicates the last execution time in $\mu$ s of the event task associated with the input %I0.3.	S
%SW132	Event execution time	Indicates the last execution time in $\mu$ s of the event task associated with the input %I0.4.	S
%SW133	Event execution time	Indicates the last execution time in $\mu$ s of the event task associated with the input %I0.5.	S
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control
%SW134	Event execution time	Indicates the last execution time in $\mu$ s of the event task associated with the Threshold 0 of HSC0 or HSC2.	S
%SW135	Event execution time	Indicates the last execution time in $\mu$ s of the event task associated with the Threshold 1 of HSC0 or HSC2.	S
%SW136	Event execution time	Indicates the last execution time in $\mu$ s of the event task associated with the Threshold 0 of HSC1 or HSC3.	S
%SW137	Event execution time	Indicates the last execution time in $\mu$ s of the event task associated with the Threshold 1 of HSC1 or HSC3.	S
%SW138	Periodic task execution time	Indicates the last execution time in $\mu$ s of the periodic task.	S
%SW139	Embedded digital output protection	Indicates the protection error status of output blocks: Bit0 = 1 - Q0 - Q3 protect error - Block0 Bit1 = 1 - Q4 - Q7 protect error - Block1 Bit2 = 1 - Q8 - Q11 protect error - Block2 Bit3 = 1 - Q12 - Q15 protect error - Block3	S
%SW148	Number of persistent variables	Maximum 1,000 variables. For more information, refer to Persistent Variables Saved by User Request ( <a href="#">see page 56</a> ).	U
%SW149	Event execution time	Indicates the last execution time in ms of the event task associated with the input %I0.2.	S
%SW150	Event execution time	Indicates the last execution time in ms of the event task associated with the input %I0.3.	S
%SW151	Event execution time	Indicates the last execution time in ms of the event task associated with the input %I0.4.	S
%SW152	Event execution time	Indicates the last execution time in ms of the event task associated with the input %I0.5.	S
%SW153	Event execution time	Indicates the last execution time in ms of the event task associated with the Threshold 0 of HSC0 or HSC2.	S
%SW154	Event execution time	Indicates the last execution time in ms of the event task associated with the Threshold 1 of HSC0 or HSC2.	S
%SW155	Event execution time	Indicates the last execution time in ms of the event task associated with the Threshold 0 of HSC1 or HSC3.	S
%SW156	Event execution time	Indicates the last execution time in ms of the event task associated with the Threshold 1 of HSC1 or HSC3.	S
%SW157	Periodic execution time	Indicates the last execution time of the periodic task in ms.	S
%SW158	Periodic average time	Indicates the average execution time in ms of the periodic task (last 5 times).	S
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control
%SW159	Event 0 average time	Indicates the average execution time in ms of the event task associated with the input %IO.2 (last 5 times).	S
%SW160	Event 1 average time	Indicates the average execution time in ms of the event task associated with the input %IO.3 (last 5 times).	S
%SW161	Event 2 average time	Indicates the average execution time in ms of the event task associated with the input %IO.4 (last 5 times).	S
%SW162	Event 3 average time	Indicates the average execution time in ms of the event task associated with the input %IO.5 (last 5 times).	S
%SW163	Event 4 average time	Indicates the average execution time in ms of the event task associated with the Threshold 0 of HSC0 or HSC2 (last 5 times).	S
%SW164	Event 5 average time	Indicates the average execution time in ms of the event task associated with the Threshold 1 of HSC0 or HSC2 (last 5 times).	S
%SW165	Event 6 average time	Indicates the average execution time in ms of the event task associated with the Threshold 0 of HSC1 or HSC3 (last 5 times).	S
%SW166	Event 7 average time	Indicates the average execution time in ms of the event task associated with the Threshold 1 of HSC1 or HSC3 (last 5 times).	S
%SW167	Status of Modem initialization command	<p>%SW167 indicates the status of the initialization command sent to the modem:</p> <ul style="list-style-type: none"> <li>● If the modem does not respond to the initialization command within 10 attempts, its value is FFFF; modem does not respond.</li> <li>● If the modem responds "OK" within the 10 attempts, its value is 0; modem is present and has accepted the initialization command.</li> <li>● If the modem sends something else within the 10 attempts, its value is 4; incorrect response from the modem, or the modem rejects the initialization command.</li> </ul> <p><b>NOTE:</b> %S105 can be used to re-send the modem initialization command.</p>	S
%SW168	Modbus TCP – Connections in use	<p>Indicates the number of Ethernet Modbus TCP server connections in use.</p> <p><b>NOTE:</b> If you disconnect the cable, the connection is not closed immediately. Each time the cable is re-connected to the network, it requests a new connection and the number of connections in use indicated by %SW168 increases.</p>	S
%SW170	Frames transmitted – Serial line 1	Indicates the count of frames transmitted by the serial line 1.	S
%SW171	Frames transmitted – Serial line 2	Indicates the count of frames transmitted by the serial line 2.	S
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>SIM</b> Applied in the simulator</p>			

System Words	Function	Description	Control
%SW172	Frames transmitted – USB	Indicates the count of frames transmitted by the USB channel.	S
%SW173	Frames transmitted – Modbus TCP	Indicates the count of frames transmitted by Modbus TCP on Ethernet.	S
%SW174	Frames received successfully – Serial line 1	Indicates the count of frames correctly received by the serial line 1.	S
%SW175	Frames received successfully – Serial line 2	Indicates the count of frames correctly received by the serial line 2.	S
%SW176	Frames received successfully – USB	Indicates the count of frames correctly received by the USB channel.	S
%SW177	Frames received successfully – Modbus TCP	Indicates the count of frames correctly received by the Modbus TCP on Ethernet.	S
%SW178	Frames received with an error – Serial line 1	Indicates the count of frames received with an error detected for the serial line 1.	S
%SW179	Frames received with an error – Serial line 2	Indicates the count of frames received with an error detected for the serial line 2.	S
%SW180	Frames received with an error – USB	Indicates the count of frames received with an error detected for the USB channel.	S
%SW181	Frames received with an error – Modbus TCP	Indicates the count of frames received with an error detected for Modbus TCP on Ethernet.	S
%SW182	Remote Graphic Display connection state	Indicates the connection state of the Remote Graphic Display: <ul style="list-style-type: none"> <li>● 0: Display not connected</li> <li>● 1: Display application not ready</li> <li>● 2: Display application transfer</li> <li>● 3: Display application running</li> <li>● 4: Display firmware update required</li> <li>● 5: Display firmware transfer in progress</li> </ul>	S
%SW183	Remote Graphic Display last error detected	Indicates the last error detected on the Remote Graphic Display: <ul style="list-style-type: none"> <li>● 0: No error detected</li> <li>● 1: Display application transfer unsuccessful</li> <li>● 2: Incompatible version of the display</li> </ul>	S
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control
%SW184	Remote Graphic Display Page Index	Indicates the page index of the displayed page on the Remote Graphic Display. When written, specifies the page index of the page to display on the Remote Graphic Display, if it exists. Otherwise, the value is ignored. A page index is generated by SoMachine Basic when the user creates a new Operator Interface page. The following pages have fixed page index values: <ul style="list-style-type: none"> <li>● 112: <b>Setup Menu</b></li> <li>● 113: <b>Controller Information</b></li> <li>● 114: <b>Controller Setup</b></li> <li>● 117: <b>Display Setup</b></li> <li>● 120: <b>Controller State</b></li> <li>● 121: <b>Controller Status</b></li> <li>● 128: <b>Alarm View</b></li> </ul>	S, U
%SW185	TMH2GDB firmware version xx.yy	Firmware version of the TMH2GDB Remote Graphic Display. For example, %SW185 = 0104 hex means that the firmware version is V1.4.	S
%SW188	Frames transmitted - Modbus Mapping table	Total number of frames transmitted via the Modbus mapping table.	S
%SW189	Frames received - Modbus Mapping table	Total number of frames received without error via the Modbus mapping table.	S
%SW190, %SW191	Class 1 outgoing packets sent	Total number of outgoing packets sent for implicit (Class 1) connections.	S
%SW192, %SW193	Class 1 incoming packets received	Total number of incoming packets received for implicit (Class 1) connections.	S
%SW194, %SW195	Unconnected incoming packets received	Total number of incoming unconnected packets, including packets that would be returned if an error was detected.	S
%SW196, %SW197	Unconnected incoming packets invalid	Total number of incoming unconnected packets that had an invalid format, or targeted an unsupported service, class, instance, attribute, or member.	S
%SW198, %SW199	Incoming packets received for explicit (Class 3) connections	Total number of incoming packets for explicit (Class 3) connections, including packets that would be returned if an error was detected.	S
%SW200, %SW201	Incoming Class 3 packets invalid	Total number of explicit (Class 3) packets that had an invalid format, or targeted an unsupported service, class, instance, attribute, or member.	S
<b>S</b> Controlled by the system <b>U</b> Controlled by the user <b>SIM</b> Applied in the simulator			

System Words	Function	Description	Control
%SW202	Instance input	Instance input configured in SoMachine Basic. Default value: 0	S
%SW203	Input size	Input size configured in SoMachine Basic. Default value: 0	S
%SW204	Instance output	Instance output configured in SoMachine Basic. Default value: 0	S
%SW205	Output size	Output size configured in SoMachine Basic. Default value: 0	S
%SW206	Timeout	Total number of connection timeouts that have occurred in connections. Default value: 0	S, U
%SW207	Status of the Ethernet/IP class 1 connection	<p>Indicates the status of the EtherNet/IP class 1 connection:</p> <ul style="list-style-type: none"> <li>● 0: At least one connection is idle.</li> <li>● 1: The open connections are in run.</li> <li>● 2: At least one connection has no indication or no communication.</li> </ul> <p><b>NOTE:</b> Status 2 overrides status 0.</p> <p><b>NOTE:</b> The application must be configured with a functional level of at least <b>Level 3.2</b> for this word to be supported.</p>	S
<p><b>S</b> Controlled by the system  <b>U</b> Controlled by the user  <b>SIM</b> Applied in the simulator</p>			

## M221 Logic Controller Code ID

This table presents the code IDs of the M221 Logic Controller references:

Reference	Code ID
TM221M16R•	0x0780
TM221ME16R•	0x0781
TM221M16T•	0x0782
TM221ME16T•	0x0783
TM221M32TK	0x0784
TM221ME32TK	0x0785
TM221C16R	0x0786
TM221CE16R	0x0787
TM221C16T	0x0788
TM221CE16T	0x0789
TM221C24R	0x078A
TM221CE24R	0x078B
TM221C24T	0x078C
TM221CE24T	0x078D
TM221C40R	0x078E
TM221CE40R	0x078F
TM221C40T	0x0790
TM221CE40T	0x0791





## A

### **analog input**

Converts received voltage or current levels into numerical values. You can store and process these values within the logic controller.

### **analog output**

Converts numerical values within the logic controller and sends out proportional voltage or current levels.

### **application**

A program including configuration data, symbols, and documentation.

## B

### **BOOTP**

*(bootstrap protocol)* A UDP network protocol that can be used by a network client to automatically obtain an IP address (and possibly other data) from a server. The client identifies itself to the server using the client MAC address. The server, which maintains a pre-configured table of client device MAC addresses and associated IP addresses, sends the client its pre-configured IP address. BOOTP was originally used as a method that enabled diskless hosts to be remotely booted over a network. The BOOTP process assigns an infinite lease of an IP address. The BOOTP service utilizes UDP ports 67 and 68.

## 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.

### **control network**

A network containing logic controllers, SCADA systems, PCs, HMI, switches, ...

Two kinds of topologies are supported:

- flat: all modules and devices in this network belong to same subnet.
- 2 levels: the network is split into an operation network and an inter-controller network.

These two networks can be physically independent, but are generally linked by a routing device.

## D

### DHCP

*(dynamic host configuration protocol)* An advanced extension of BOOTP. DHCP is more advanced, but both DHCP and BOOTP are common. (DHCP can handle BOOTP client requests.)

### digital I/O

*(digital input/output)* An individual circuit connection at the electronic module that corresponds directly to a data table bit. The data table bit holds the value of the signal at the I/O circuit. It gives the control logic digital access to I/O values.

## E

### EDS

*(electronic data sheet)* A file for fieldbus device description that contains, for example, the properties of a device such as parameters and settings.

### EtherNet/IP

*(Ethernet industrial protocol)* An open communications protocol for manufacturing automation solutions in industrial systems. EtherNet/IP is in a family of networks that implement the common industrial protocol at its upper layers. The supporting organization (ODVA) specifies EtherNet/IP to accomplish global adaptability and media independence.

### EtherNet/IP Adapter

An EtherNet/IP Adapter, sometimes also called a server, is an end-device in an EtherNet/IP network. I/O blocks and drives can be EtherNet/IP Adapter devices.

### expansion bus

An electronic communication bus between expansion I/O modules and a controller.

## 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.

## H

### HMI

*(human machine interface)* An operator interface (usually graphical) for human control over industrial equipment.

**I****I/O**

(*input/output*)

**IEC 61131-3**

Part 3 of a 3-part IEC standard for industrial automation equipment. IEC 61131-3 is concerned with controller programming languages and defines 2 graphical and 2 textual programming language standards. The graphical programming languages are ladder diagram and function block diagram. The textual programming languages include structured text and instruction list.

**IL**

(*instruction list*) A program written in the 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 (refer to IEC 61131-3).

**Input Assembly**

Assemblies are blocks of data exchanged between network devices and the logic controller. An Input Assembly generally contains status information from a network device read by the controller.

**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).

**LAN**

(*local area network*) A short-distance communications network that is implemented in a home, office, or institutional environment.

**LD**

(*ladder diagram*) 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 (refer to IEC 61131-3).

**LSB**

(*least significant bit/byte*) The part of a number, address, or field that is written as the right-most single value in conventional hexadecimal or binary notation.

## 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.

### **Modbus**

The protocol that allows communications between many devices connected to the same network.

### **MSB**

*(most significant bit/byte)* The part of a number, address, or field that is written as the left-most single value in conventional hexadecimal or binary notation.

## O

### **Output Assembly**

Assemblies are blocks of data exchanged between network devices and the logic controller. An Output Assembly generally contains command sent by the controller to network devices.

## P

### **periodic execution**

The task is executed either cyclically or periodically. In periodic mode, you determine a specific time (period) in which the task is executed. If it is executed under this time, a waiting time is generated before the next cycle. If it is executed over this time, a control system indicates the overrun. If the overrun is too high, the controller is stopped.

### **periodic task**

The periodic task is a periodic, high-priority task of short duration that runs on a logic controller through its programming software. The short duration of the periodic task prevents it from interfering with the execution of slower, lower priority tasks. A periodic task is useful when fast periodic changes in digital inputs need to be monitored.

### **PID**

*(proportional, integral, derivative)* A generic control loop feedback mechanism (controller) widely used in industrial control systems.

### **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.

### **program**

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

**protocol**

A convention or standard definition that controls or enables the connection, communication, and data transfer between 2 computing system and devices.

**R****RTC**

*(real-time clock)* A battery-backed time-of-day and calendar clock that operates continuously, even when the controller is not powered for the life of the battery.

**S****security parameters**

A set of configuration parameters used to enable or disable specific protocols and features relating to the cybersecurity of an application.

**SFC**

*(sequential function chart)* A language that is composed of steps with associated actions, transitions with associated logic condition, and directed links between steps and transitions. (The SFC standard is defined in IEC 848. It is IEC 61131-3 compliant.)

**SMS**

*(short message service)* A standard communication service for telephones (or other devices) that send short text messages over the mobile communications system.





## Symbols

%C, 36  
%DR, 36  
%FC, 36  
%HSC, 36  
%I, 36, 164  
%IW, 36, 166  
%IWE, 36, 172  
%IWM, 36, 175  
%KD, 36  
%KF, 36  
%KW, 36  
%M, 36  
%MD, 36  
%MF, 36  
%MSG, 36  
%MW, 36  
%PLS, 36  
%PWM, 36  
%Q, 36, 165  
%QW, 36, 168  
%QWE, 36, 170  
%QWM, 36, 173  
%R, 36  
%S, 36  
%S (system bits), 180  
%S93, 56  
%S94, 56  
%SBR, 36  
%SC, 36  
%SW, 36  
%SW (system words), 188  
%SW118, 75  
%SW119, 75  
%SW120, 75  
%SW148, 56, 56  
%SW6, 52  
%TM, 36

## A

advanced expert functions, 177  
analog inputs, 86  
    configuration, 86  
    introduction, 86  
    properties, 166  
analog outputs  
    properties, 168  
application download, 53

## B

backup controller memory, 159  
boot controller, 53

## C

cartridges  
    configuration, 117  
    TMC2, 117  
Cold Start, 55  
configuration  
    building a configuration, 68  
    configuration introduction, 68  
    Frequency Meter, 96  
    HSC, 92  
controller  
    configuration, 67, 77  
    configuration features, 29  
controller state, 48, 49  
    BOOTING, 51  
    EMPTY, 51  
    HALTED, 52  
    POWERLESS, 52  
    RUNNING, 51  
    STOPPED, 51

**D**

- digital inputs, 80
  - configuration, 80
  - introduction, 80
  - properties, 164
- digital outputs, 84
  - configuration, 84
  - configuration parameters, 84
  - configuring fallback values for, 84
  - introduction, 84
  - properties, 165
- downloading applications, 52

**E**

- EDS file, Modbus TCP, 132
- embedded communication
  - configuration, 119
- embedded input/output
  - configuration, 79
- Ethernet
  - configuration, 122
  - introduction, 121
- Ethernet services, 121
- Ethernet/IP
  - configuration, 130
- executive loader, 78
- expansion modules
  - configuration, 117
  - TM2, 117
  - TM3, 117

**F**

- fallback
  - values, configuring, 84
- fallback behavior configuration, 58
- fallback execution, 58
- fallback management, 57
- fallback values, 58
- fallbackj value, 170, 173
- features
  - key features, 20, 25

- firmware, 78, 150
  - updating with executive loader, 78
  - updating with SD card, 150
- firmware updates, 52
- Frequency Meter
  - configuration, 96

**H**

- HALTED state, 54
- hardware initialization values, 57
- high speed counters, 89
  - configuration, 90
  - introduction, 89
- HSC
  - configuration, 92

**I**

- I/O assignment, 89
- I/O bus
  - configuration, 111
- I/O configuration general information
  - general practices, 112
- I/O objects
  - analog inputs, 166
  - Analog Outputs, 168
  - digital inputs, 164
  - digital outputs, 165
- Init command, 134
- initialization values, 57
- Initialize controller, 53
- Input assembly
  - properties, 170
- input registers
  - properties, 173

**M**

- Machine.cfg (post configuration file), 63
- maintain values fallback mode, 170, 173
- mapping table, modbus TCP, 128, 142, 173, 175
- memory objects
  - backing up and restoring, 159

modbus mapping table, 126, 128  
modbus TCP  
  configuration, 126  
  EDS file, 132  
  mapping table, 142, 173  
  remote servers, 128

## N

network objects, 169  
  Input assembly, 170  
  Input registers, 173  
  Output assembly, 172  
  Output registers, 175

## O

objects  
  addressing, 36  
  addressing examples, 36  
  definition of, 31  
  introduction, 32  
  maximum number allowed, 38  
  network, 169  
  object types, 32  
Output assembly  
  properties, 172  
output behavior, 57, 59  
output forcing, 59  
output rearming, 60  
output registers  
  properties, 175

## P

persistent variables, 56  
Post Conf  
  Presentation, 62  
post configuration  
  file management, 63  
Post Configuration  
  Presentation, 62  
programming languages  
  IL, LD, 25  
  IL, LD, Grafcet, 20

pulse generators, 99  
  configuration, 99  
  FREQGEN configuration, 109  
  introduction, 99  
  PLS configuration, 102  
  PTO configuration, 106  
  PWM configuration, 104

## R

rearming outputs, 60  
restore controller memory, 159  
RUN Controller, 54  
Run/Stop, 82  
  configuring digital input as, 82

## S

SD card, 150  
  application management, 152  
  cloning, 148  
  post configuration management, 154  
  updating firmware, 150  
serial line, 133  
  configuration, 134  
  configuring to use %SEND\_RECV\_SMS,  
  134  
  introduction, 133  
software initialization values, 57  
STOP Controller, 54  
supported devices, 117  
system bits  
  %S93, 56  
  %S94, 56  
system bits (%S), 180  
system words  
  %SW118, 75  
  %SW119, 75  
  %SW120, 75  
  %SW148, 56, 56

## U

updating firmware, 78, 150  
uploading applications, 52

**W**

Warm start, 55