Modicon M221
Logic Controller
Programming Guide

12/2017

SoMachine Basic

EIO0000001360.09

www.schneider-electric.com

Schneider Electric

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

You agree not to reproduce, other than for your own personal, noncommercial use, all or part of this document on any medium whatsoever without permission of Schneider Electric, given in writing. You also agree not to establish any hypertext links to this document or its content. Schneider Electric does not grant any right or license for the personal and noncommercial use of the document or its content, except for a non-exclusive license to consult it on an "as is" basis, at your own risk. All other rights are reserved.

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.

Table of Contents

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 V1.6.

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

Step	Action
1	Go to the Schneider Electric home page *www.schneider-electric.com*.
2	In the **Search** box type the reference of a product or the name of a product range. ● Do not include blank spaces in the reference or product range. ● To get information on grouping similar modules, use asterisks (*).
3	If you entered a reference, go to the **Product Datasheets** search results and click on the reference that interests you. If you entered the name of a product range, go to the **Product Ranges** search results and click on the product range that interests you.
4	If more than one reference appears in the **Products** search results, click on the reference that interests you.
5	Depending on the size of your screen, you may need to scroll down to see the data sheet.
6	To save or print a data sheet as a .pdf file, click **Download XXX product datasheet**.

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

Related Documents

Title of Documentation	Reference Number
SoMachine Basic - 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 (FRE)* *EIO0000001476 (GER)* *EIO0000001477 (SPA)* *EIO0000001478 (ITA)* *EIO0000001479 (CHS)* *EIO0000001480 (POR)* *EIO0000001481 (TUR)*
Modicon M221 Logic Controller Advanced Functions - Library Guide	*EIO0000002007 (ENG)* *EIO0000002008 (FRE)* *EIO0000002009 (GER)* *EIO0000002010 (SPA)* *EIO0000002011 (ITA)* *EIO0000002012 (CHS)* *EIO0000002013 (POR)* *EIO0000002014 (TUR)*
Modicon M221 Logic Controller - Hardware Guide	*EIO0000001384 (ENG)* *EIO0000001385 (FRE)* *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 (FRE)* *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)* *EIO0000001774 (POR)* *EIO0000001775 (TUR)*
Modicon TM3 Expansion Modules Configuration - Programming Guide	*EIO0000001396 (ENG)* *EIO0000001397 (FRE)* *EIO0000001398 (GER)* *EIO0000001399 (SPA)* *EIO0000001400 (ITA)* *EIO0000001401 (CHS)* *EIO0000001374 (POR)* *EIO0000001375 (TUR)*
Modicon TM3 Digital I/O Modules - Hardware Guide	*EIO0000001408 (ENG)* *EIO0000001409 (FRE)* *EIO0000001410 (GER)* *EIO0000001411 (SPA)* *EIO0000001412 (ITA)* *EIO0000001413 (CHS)* *EIO0000001376 (POR)* *EIO0000001377 (TUR)*
Modicon TM3 Analog I/O Modules - Hardware Guide	*EIO0000001414 (ENG)* *EIO0000001415 (FRE)* *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 (FRE)* *EIO0000001422 (GER)* *EIO0000001423 (SPA)* *EIO0000001424 (ITA)* *EIO0000001425 (CHS)* *EIO0000001380 (POR)* *EIO0000001381 (TUR)*
Modicon TM3 Safety Modules - Hardware Guide	*EIO0000001831 (ENG)* *EIO0000001832 (FRE)* *EIO0000001833 (GER)* *EIO0000001834 (SPA)* *EIO0000001835 (ITA)* *EIO0000001836 (CHS)* *EIO0000001837 (POR)* *EIO0000001838 (TUR)*
Modicon TM3 Transmitter and Receiver Modules - Hardware Guide	*EIO0000001426 (ENG)* *EIO0000001427 (FRE)* *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 (FRE)* *EIO0000000030 (GER)* *EIO0000000031 (SPA)* *EIO0000000032 (ITA)* *EIO0000000033 (CHS)*

Title of Documentation	Reference Number
Modicon TM2 Analog I/O Modules - Hardware Guide	*EIO0000000034 (ENG)* *EIO0000000035 (FRE)* *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://www.schneider-electric.com/en/download

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

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.
IEC 61784-3:2008	Digital data communication for measurement and control: Functional safety field buses.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	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 *Machinery Directive* (*2006/42/EC*) 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	31

Chapter 1
About the Modicon M221 Logic Controller

What Is in This Chapter?

This chapter contains the following topics:

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 *(see 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 TM221C Logic Controller is 24 Vdc *(see Modicon M221 Logic Controller, Hardware Guide)* or 100...240 Vac *(see Modicon M221 Logic Controller, Hardware Guide)*.

Real Time Clock

The M221 Logic Controller includes a Real Time Clock (RTC) system *(see Modicon M221 Logic Controller, Hardware Guide)*.

Run/Stop

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

- a hardware Run/Stop switch *(see Modicon M221 Logic Controller, Hardware Guide)*
- a Run/Stop *(see Modicon M221 Logic Controller, Hardware Guide)* operation by a dedicated digital input, defined in the software configuration (for more information, refer to Configuring Digital Inputs *(see page 90)*.)
- 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 *(see Modicon M221 Logic Controller, Hardware Guide)*.

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

- Clone management *(see page 202)*: back up the application, firmware, and post configuration (if it exists) of the logic controller
- Firmware management *(see page 204)*: download firmware to the logic controller, to a TMH2GDB Remote Graphic Display , or to TM3 expansion modules
- Application management *(see page 208)*: back up and restore the logic controller application, or copy it to another logic controller of the same reference
- Post configuration management *(see page 210)*: add, change, or delete the post configuration file of the logic controller
- Error log management *(see page 212)*: back up or delete the error log file of the logic controller
- Memory management *(see page 215)*: back up and restore memory bits and words from a controller

Embedded Communication Features

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

- Ethernet *(see Modicon M221 Logic Controller, Hardware Guide)*
- USB Mini-B *(see Modicon M221 Logic Controller, Hardware Guide)*
- Serial Line 1 *(see Modicon M221 Logic Controller, Hardware Guide)*

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 *(see Modicon M221 Logic Controller, Hardware Guide)*	5 regular inputs(1) 4 fast inputs (HSC)(2)	7 relay outputs	Yes	1 serial line port 1 USB programming port	100...240 Vac
TM221CE16R *(see Modicon M221 Logic Controller, Hardware Guide)*			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C16T *(see Modicon M221 Logic Controller, Hardware Guide)*	5 regular inputs(1) 4 fast inputs (HSC)(2)	Source outputs 5 regular transistor outputs 2 fast outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE16T *(see Modicon M221 Logic Controller, Hardware Guide)*			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C16U	5 regular inputs(1) 4 fast inputs (HSC)(2)	Sink outputs 5 regular transistor outputs 2 fast outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE16U *(see Modicon M221 Logic Controller, Hardware Guide)*				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, for PLS, PWM, PTO, or FREQGEN functions, or reflex outputs for HSC.

Reference	Digital Inputs	Digital Outputs	Analog Inputs	Communication Ports	Power Supply
TM221C24R *(see Modicon M221 Logic Controller, Hardware Guide)*	10 regular inputs(1) 4 fast inputs (HSC)(2)	10 relay outputs	Yes	1 serial line port 1 USB programming port	100...240 Vac
TM221CE24R *(see Modicon M221 Logic Controller, Hardware Guide)*			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C24T *(see Modicon M221 Logic Controller, Hardware Guide)*		Source outputs 8 regular transistor outputs 2 fast outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE24T *(see Modicon M221 Logic Controller, Hardware Guide)*			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C24U	10 regular inputs(1) 4 fast inputs (HSC)(2)	Sink outputs 8 regular transistor outputs 2 fast outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE24U *(see Modicon M221 Logic Controller, Hardware Guide)*			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, for PLS, PWM, PTO, or FREQGEN functions, or reflex outputs for HSC.

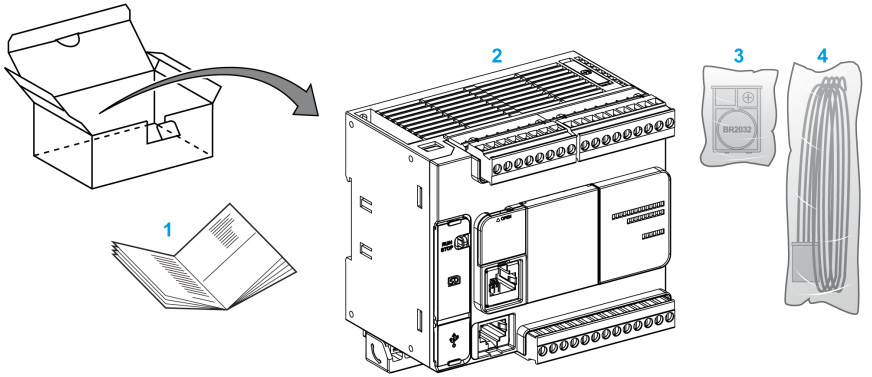
Reference	Digital Inputs	Digital Outputs	Analog Inputs	Communication Ports	Power Supply
TM221C40R *(see Modicon M221 Logic Controller, Hardware Guide)*	20 regular inputs(1) 4 fast inputs (HSC)(2)	16 relay outputs	Yes	1 serial line port 1 USB programming port	100...240 Vac
TM221CE40R *(see Modicon M221 Logic Controller, Hardware Guide)*			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C40T *(see Modicon M221 Logic Controller, Hardware Guide)*		Source outputs 14 regular transistor outputs 2 fast outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE40T *(see Modicon M221 Logic Controller, Hardware Guide)*			Yes	1 serial line port 1 USB programming port 1 Ethernet port	
TM221C40U *(see Modicon M221 Logic Controller, Hardware Guide)*	20 regular inputs(1) 4 fast inputs (HSC)(2)	Sink outputs 12 regular transistor outputs 4 fast outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	1 serial line port 1 USB programming port	24 Vdc
TM221CE40U *(see Modicon M221 Logic Controller, Hardware Guide)*			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, for PLS, PWM, PTO, or FREQGEN 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 *(see 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 *(see Modicon M221 Logic Controller, Hardware Guide)*.

Real Time Clock

The M221 Logic Controller includes a Real Time Clock (RTC) system *(see Modicon M221 Logic Controller, Hardware Guide)*.

Run/Stop

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

- a hardware Run/Stop switch *(see Modicon M221 Logic Controller, Hardware Guide)*
- a Run/Stop *(see Modicon M221 Logic Controller, Hardware Guide)* operation by a dedicated digital input, defined in the software configuration (for more information, refer to Configuring Digital Inputs *(see page 90)*)
- 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/FREQGEN)
- Relay outputs
- Analog inputs

Removable Storage

The M221 Logic Controllers include an embedded SD card slot *(see Modicon M221 Logic Controller, Hardware Guide)*.

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

- Clone management *(see page 202)*: back up the application, firmware, and post configuration (if it exists) of the logic controller
- Firmware management *(see page 204)*: download firmware updates directly to the logic controller, and download firmware to a TMH2GDB Remote Graphic Display
- Application management *(see page 208)*: back up and restore the logic controller application, or copy it to another logic controller of the same reference
- Post configuration management *(see page 210)*: add, change, or delete the post configuration file of the logic controller
- Error log management *(see page 212)*: back up or delete the error log file of the logic controller
- Memory management *(see page 215)*: 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 *(see Modicon M221 Logic Controller, Hardware Guide)*
- USB Mini-B *(see Modicon M221 Logic Controller, Hardware Guide)*
- SD Card *(see Modicon M221 Logic Controller, Hardware Guide)*
- Serial Line 1 *(see Modicon M221 Logic Controller, Hardware Guide)*
- Serial Line 2 *(see Modicon M221 Logic Controller, Hardware Guide)*

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 *(see Modicon M221 Logic Controller, Hardware Guide)*	4 regular inputs(1) 4 fast inputs (HSC)(2)	8 relay outputs	Yes	2 serial line ports 1 USB programming port	Removable screw terminal blocks
TM221M16RG *(see Modicon M221 Logic Controller, Hardware Guide)*	4 regular inputs(1) 4 fast inputs (HSC)(2)	8 relay outputs	Yes	2 serial line ports 1 USB programming port	Removable spring terminal blocks
TM221ME16R *(see Modicon M221 Logic Controller, Hardware Guide)*	4 regular inputs(1) 4 fast inputs (HSC)(2)	8 relay outputs	Yes	1 serial line port 1 USB programming port 1 Ethernet port	Removable screw terminal blocks
TM221ME16RG *(see Modicon M221 Logic Controller, Hardware Guide)*	4 regular inputs(1) 4 fast inputs (HSC)(2)	8 relay outputs	Yes	1 serial line port 1 USB programming port 1 Ethernet port	Removable spring terminal blocks
TM221M16T *(see Modicon M221 Logic Controller, Hardware Guide)*	4 regular inputs(1) 4 fast inputs (HSC)(2)	6 regular transistor outputs 2 fast transistor outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	2 serial line ports 1 USB programming port	Removable screw terminal blocks

NOTE: The TM221M Logic Controller uses a 24 Vdc power supply *(see Modicon M221 Logic Controller, Hardware Guide)*.

(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, for PLS, PWM, PTO or FREQGEN functions, or reflex outputs for HSC.

Reference	Digital Input	Digital Output	Analog Input	Communication Ports	Terminal Type
TM221M16TG *(see Modicon M221 Logic Controller, Hardware Guide)*	4 regular inputs(1) 4 fast inputs (HSC)(2)	6 regular transistor outputs 2 fast transistor outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	2 serial line ports 1 USB programming port	Removable spring terminal blocks
TM221ME16T *(see Modicon M221 Logic Controller, Hardware Guide)*	4 regular inputs(1) 4 fast inputs (HSC)(2)	6 regular transistor outputs 2 fast transistor outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	1 serial line port 1 USB programming port 1 Ethernet port	Removable screw terminal blocks
TM221ME16TG *(see Modicon M221 Logic Controller, Hardware Guide)*	4 regular inputs(1) 4 fast inputs (HSC)(2)	6 regular transistor outputs 2 fast transistor outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	1 serial line port USB programming port 1 Ethernet port	Removable spring terminal blocks
TM221M32TK *(see Modicon M221 Logic Controller, Hardware Guide)*	12 regular inputs(1) 4 fast inputs (HSC)(2)	14 regular transistor outputs 2 fast outputs (PLS/PWM/PTO/FREQGEN)(3)	Yes	2 serial line ports 1 USB programming port	HE10 (MIL 20) connectors
TM221ME32TK *(see Modicon M221 Logic Controller, Hardware Guide)*	12 regular inputs(1) 4 fast inputs (HSC)(2)	14 regular outputs 2 fast outputs (PLS/PWM/PTO/FREQGEN)(3)	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 *(see Modicon M221 Logic Controller, Hardware Guide)*.

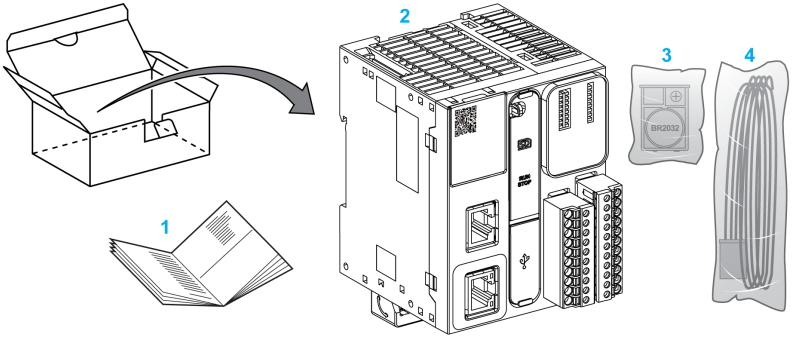
(1) The regular inputs I2, I3, I4, and I5 have a maximum frequency of 5 kHz.
The other regular inputs have a maximum frequency of 100 Hz.

(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, for PLS, PWM, PTO or FREQGEN 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 2.1
Objects

What Is in This Section?

This section contains the following topics:

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 *(see SoMachine Basic, Generic Functions Library Guide)*	Stores memory bit.
	%MW	Memory words *(see SoMachine Basic, Generic Functions Library Guide)*	Stores 16-bit memory word.
	%MD	Memory double words *(see SoMachine Basic, Generic Functions Library Guide)*	Stores 32-bit memory word.
	%MF	Memory floating point *(see SoMachine Basic, Generic Functions Library Guide)*	Stores memory floating point in a mathematical argument which has a decimal in its expression.
	%KW	Constant words *(see SoMachine Basic, Generic Functions Library Guide)*	Stores 16-bit constant word.
	%KD	Constant double words *(see SoMachine Basic, Generic Functions Library Guide)*	Stores 32-bit constant word.
	%KF	Constant floating points *(see SoMachine Basic, Generic Functions Library Guide)*	Stores constant floating point in a mathematical argument which has a decimal in its expression.
System objects	%S	System bits *(see page 242)*	Stores system bit.
	%SW	System words *(see page 254)*	Stores system word.
	%IWS	Input channel status word *(see page 280)*	Contains diagnostic information concerning analog input channels.
	%QWS	Output channel status word *(see page 282)*	Contains diagnostic information concerning analog output channels.

Object Type	Object	Object Function	Description
I/O objects	%I	Input bits *(see page 220)*	Stores value of the digital input.
	%Q	Output bits *(see page 221)*	Stores value of the digital output.
	%IW	Input words *(see page 222)*	Stores value of the analog input.
	%QW	Output words *(see page 224)*	Stores value of the analog output.
	%FC	Fast counters *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*	Execute fast counts of pulses from sensors, switches, and so on.
	%HSC	High speed counters *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*	Execute fast counts of pulses from sensors, switches, and so on, that are connected to the fast inputs.
	%PLS	Pulse *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*	Generates a square wave pulse signal on dedicated output channels.
	%PWM	Pulse width modulation *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*	Generates a modulated wave signal on dedicated output channels with a variable duty cycle.
	%PTO	Pulse train output *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*	Generates a pulse train output to control a linear single-axis stepper or servo drive in open loop mode.
	%FREQGEN	Frequency Generator *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*	Generates a square wave signal on a dedicated output channel with programmable frequency and duty cycle of 50%.

Object Type	Object	Object Function	Description
Network objects	%QWE	Input assembly (EtherNet/IP) *(see page 226)*	The values of EtherNet/IP Input assembly frames sent by the logic controller. **NOTE:** For more details about directionality, refer to Configuring EtherNet/IP *(see page 158)*.
	%IWE	Output assembly (EtherNet/IP) *(see page 228)*	The values of EtherNet/IP Output assembly frames received by the logic controller. **NOTE:** For more details about directionality, refer to Configuring EtherNet/IP *(see page 158)*.
	%QWM	Input registers (Modbus TCP) *(see page 229)*	The values of Modbus mapping table Input registers sent by the logic controller.
	%IWM	Output registers (Modbus TCP) *(see page 231)*	The values of Modbus mapping table Output registers received by the logic controller.
	%IN	Digital inputs (IOScanner) *(see page 232)*	The values of Modbus Serial or TCP IOScanner digital input bits.
	%QN	Digital outputs (IOScanner) *(see page 234)*	The values of Modbus Serial or TCP IOScanner digital output bits.
	%IWN	Input registers (IOScanner) *(see page 236)*	The values of Modbus Serial or TCP IOScanner digital input words.
	%QWN	Output registers (IOScanner) *(see page 238)*	The values of Modbus Serial or TCP IOScanner digital output words.
	%IWNS	IOScanner network diagnostic codes *(see page 240)*	The values of Modbus Serial or TCP IOScanner network diagnostic bits.

Object Type	Object	Object Function	Description
Software objects	%TM	Timers *(see SoMachine Basic, Generic Functions Library Guide)*	Specifies a time before triggering an action.
	%C	Counters *(see SoMachine Basic, Generic Functions Library Guide)*	Provides up and down counting of actions.
	%MSG	Messages *(see SoMachine Basic, Generic Functions Library Guide)*	Stores the status message at the communication port.
	%R	LIFO/FIFO registers *(see SoMachine Basic, Generic Functions Library Guide)*	Stores memory up to 16 words of16 bits each in 2 different ways, queue, and stacks.
	%DR	Drums *(see SoMachine Basic, Generic Functions Library Guide)*	Operates on a principle similar to an electromechanical drum controller which changes step according to external events.
	%SBR	Shift bit registers *(see SoMachine Basic, Generic Functions Library Guide)*	Provides a left or right shift of binary data bits (0 or 1).
	%SC	Step counters *(see SoMachine Basic, Generic Functions Library Guide)*	Provides a series of steps to which actions can be assigned.
	SCH	Schedule blocks *(see SoMachine Basic, Generic Functions Library Guide)*	Controls actions at a predefined month, day, and time.
	%RTC	RTC *(see SoMachine Basic, Generic Functions Library Guide)*	Allows reading or writing the value of the Real Time Clock (RTC) on the logic controller.
	PID	PID *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*	Provides a generic control loop feedback in which output is proportional, integral, and derivative of the input.
	%X	Grafcet steps *(see SoMachine Basic, Generic Functions Library Guide)*	Bit objects associated with individual Grafcet (SFC) steps. Object is set to 1 when the corresponding step is active, and set to 0 when the step is deactivated.

Object Type	Object	Object Function	Description
PTO objects	Refer to Pulse Train Output *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*.		
Drive objects	Refer to Drive Objects *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*.		
Communication objects	%READ_VAR	Read Var *(see SoMachine Basic, Generic Functions Library Guide)*	The %READ_VAR function block is used to read data from a remote device on Modbus SL or Modbus TCP.
	%WRITE_VAR	Write Var *(see SoMachine Basic, Generic Functions Library Guide)*	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 *(see SoMachine Basic, Generic Functions Library Guide)*	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 *(see SoMachine Basic, Generic Functions Library Guide)*	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 *(see SoMachine Basic, Generic Functions Library Guide)*	The %SEND_RECV_SMS function block is used to send or receive SMS messages through a GSM modem connected to a serial line.
User-defined function and user-defined function block objects	%RET0	Return value *(see SoMachine Basic, Operating Guide)*	The return value of a user-defined function.
	%PARAM	Parameter *(see SoMachine Basic, Operating Guide)*	Parameters of a user-defined function or user-defined function block. The parameters are different for each object type.
	%VAR	Local variable *(see SoMachine Basic, Operating Guide)*	Local variables of a user-defined function or user-defined function block. The local variables are different for each object type.

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 217)* 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, Drive, and PTO objects, refer to the Advanced Functions Library Guide.

For more information on user-defined functions and user-defined function blocks, refer to SoMachine Basic Operating Guide *(see SoMachine Basic, Operating Guide)*.

Addressing I/O Objects

Addressing Examples

This table presents addressing examples for various object types:

Object Type	Syntax	Example	Description
Memory objects			
Memory bits	%M*i*	%M25	Internal memory bit 25.
Memory words	%MW*i*	%MW15	Internal memory word 15.
Memory double words	%MD*i*	%MD16	Internal memory double word 16.
Memory floating points	%MF*i*	%MF17	Internal memory floating point 17.
Constant words	%KW*i*	%KW26	Constant word 26.
Constant double words	%KD*i*	%KD27	Internal constant double word 27.
Constant floating points	%KF*i*	%KF28	Internal constant floating point 28.
System objects			
System bits	%S*i*	%S8	System bit 8.
System words	%SW*i*	%SW30	System word 30.
I/O objects			
Digital inputs	%I*y.z*	%I0.5	Digital input 5 on the controller (embedded I/O).
Digital outputs	%Q*y.z*	%Q3.4	Digital output 4 on the expansion module at address 3 (expansion module I/O).
Analog inputs	%IW*y.z*	%IW0.1	Analog input 1 on the controller (embedded I/O).
Analog outputs	%QW0.*m*0*n*	%QW0.100	Analog output 0 on the cartridge 1.
Fast counters	%FC*i*	%FC2	Fast counter 2 on the controller.
High speed counters	%HSC*i*	%HSC1	High speed counter 1 on the controller.

a 100 + device number on SL1, 200 + device number on SL2, 300 + device number on ETH1.
b Channel number of the Modbus Serial IOScanner or Modbus TCP IOScanner device.
c Object instance identifer in the channel.
i Object instance identifier that indicates the instance of the object on the controller.
m Cartridge number on the controller.
n Channel number on the cartridge.
y Indicates the I/O type. It is 0 for the controller and 1, 2, and so on, for the expansion modules.
z Channel number on the controller or expansion module.

Object Type	Syntax	Example	Description
Pulse	%PLS*i*	%PLS0	Pulse output 0 on the controller.
Pulse width modulation	%PWM*i*	%PWM1	Pulse width modulation output 1 on the controller.
Pulse train output	%PTO*i*	%PTO1	Pulse train output 1 on the controller.
Frequency generator	%FREQGEN*i*	%FREQGEN1	Frequency generator 1 on the controller.
Network objects			
Input assembly (EtherNet/IP)	%QWE*i*	%QWE8	Input assembly instance 8.
Output assembly (EtherNet/IP)	%IWE*i*	%IWE6	Output assembly instance 6.
Input registers (Modbus TCP)	%QWM*i*	%QWM1	Input register instance 1.
Output registers (Modbus TCP)	%IWM*i*	%IWM0	Output register instance 0.
Digital inputs (IOScanner)	%IN*a.b.c*	%IN300.2.1	Modbus TCP IOScanner slave device 0 on ETH1, channel 2, digital input 1.
Digital outputs (IOScanner)	%QN*a.b.c*	%QN101.1.0	Modbus Serial IOScanner slave device 1 on SL1, channel 1, digital output 0.
Input registers (IOScanner)	%IWN*a.b.c*	%IWN302.3.0	Modbus TCP IOScanner slave device 2 on ETH1, channel 3, input register 0.
Output registers (IOScanner)	%QWN*a.b.c*	%QWN205.0.4	Modbus Serial IOScanner slave device 5 on SL2, channel 0, output register 4.
IOScanner network diagnostic codes	%IWNS*a*	%IWNS302	Status of Modbus TCP IOScanner slave device 2 on ETH1.
	%IWNS*a.b*	%IWNS205.3	Status of channel 3 of Modbus Serial IOScanner slave device 5 on serial line SL2

a 100 + device number on SL1, 200 + device number on SL2, 300 + device number on ETH1.
b Channel number of the Modbus Serial IOScanner or Modbus TCP IOScanner device.
c Object instance identifer in the channel.
i Object instance identifier that indicates the instance of the object on the controller.
m Cartridge number on the controller.
n Channel number on the cartridge.
y Indicates the I/O type. It is 0 for the controller and 1, 2, and so on, for the expansion modules.
z Channel number on the controller or expansion module.

Object Type	Syntax	Example	Description
Software objects			
Timers	%TM*i*	%TM5	Timer instance 5.
Counters	%C*i*	%C2	Counter instance 2.
Message	%MSG*i*	%MSG1	Program compilation status message 1.
LIFO/FIFO registers	%R*i*	%R3	FIFO/LIFO registers instance 3.
Drums	%DR*i*	%DR6	Drum register 6 on the controller.
Shift bit registers	%SBR*i*	%SBR5	Shift bit register 5 on the controller.
Step counters	%SC*i*	%SC5	Step counter 5 on the controller.
Schedule blocks	SCH *i*	SCH 3	Schedule block 3 on the controller.
RTC	RTC*i*	RTC 1	Real-time clock (RTC) instance 1.
PID	PID *i*	PID 7	PID feedback object 7 on the controller.
Grafcet Steps	X*i*	X1	Grafcet step 1.
PTO objects			
MC_Power_PTO (motion function block)	%MC_POWER_PTO*i*	%MC_POWER_PTO1	MC_POWER_PTO function block instance 1.
MC_Reset_PTO (administrative function block)	%MC_RESET_PTO*i*	%MC_RESET_PTO0	MC_RESET_PTO function block instance 0.
Communication objects			
Read Var	%READ_VAR*i*	%READ_VAR2	READ_VAR function block instance 2.
Write Var	%WRITE_VAR*i*	%WRITE_VAR4	WRITE_VAR function block instance 4.
Write Read Var	%WRITE_READ_VAR*i*	%WRITE_READ_VAR0	WRITE_READ_VAR function block instance 0.

a 100 + device number on SL1, 200 + device number on SL2, 300 + device number on ETH1.
b Channel number of the Modbus Serial IOScanner or Modbus TCP IOScanner device.
c Object instance identifer in the channel.
i Object instance identifier that indicates the instance of the object on the controller.
m Cartridge number on the controller.
n Channel number on the cartridge.
y Indicates the I/O type. It is 0 for the controller and 1, 2, and so on, for the expansion modules.
z Channel number on the controller or expansion module.

Object Type	Syntax	Example	Description
Send Receive Message	%SEND_RECV_MSG*i*	%SEND_RECV_MSG6	SEND_RECV_MSG function block instance 6.
Send Receive SMS	%SEND_RECV_SMS*i*	%SEND_RECV_SMS0	SEND_RECV_SMS function block instance 0.
User-defined function and user-defined function block objects			
Return value	%RET*i*	%RET0	Return value of a user-defined function.
Parameters	%PARAM*i*	%PARAM0	Parameter of a user-defined function.
Local variables	%VAR*i*	%VAR0	Local variables of a user-defined function.

a 100 + device number on SL1, 200 + device number on SL2, 300 + device number on ETH1.
b Channel number of the Modbus Serial IOScanner or Modbus TCP IOScanner device.
c Object instance identifer in the channel.
i Object instance identifier that indicates the instance of the object on the controller.
m Cartridge number on the controller.
n Channel number on the cartridge.
y Indicates the I/O type. It is 0 for the controller and 1, 2, and so on, for the expansion modules.
z Channel number on the controller or expansion module.

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
Memory objects				
%M(1)	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
System objects				
%S	160	160	160	160
%SW	234	234	234	234
%IWS	1 created automatically for each analog input			
%QWS	1 created automatically for each analog output			
I/O objects				
%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•)

(1) The value 512 is for 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•)	7 (for TM221C16• and TM221CE16•)	7 (for TM221C16• and TM221CE16•)
		16 (for TM221M32TK and TM221ME32TK)	10 (for TM221C24• and TM221CE24•)	10 (for TM221C24• and TM221CE24•)
			16 (for TM221C40• and TM221CE40•)	16 (for TM221C40• and TM221CE40•)
%IW	2	2	2	2
%QW	0	0	**NOTE:** 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)	2 (if 1 cartridge is used) 4 (if 2 cartridges are used with TM221C40T or TM221CE40T or TM221C••U or TM221CE••U)
%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
Network objects				
%QWE	20 (for TM221ME16R•)	20 (for TM221ME16T• and TM221ME32TK)	20 (for TM221CE16•)	20 (for TM221CE16•)
(1) The value 512 is for 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•)
%IN	128	128	128	128
%QN	128	128	128	128
%IWN	128	128	128	128
%QWN	128	128	128	128
%IWNS	1 for each configured Modbus Serial IOScanner or Modbus TCP IOScanner device, plus 1 for each channel			
%QWNS	1 for each configured Modbus Serial IOScanner or Modbus TCP IOScanner device, plus 1 for each channel			
Software objects				
%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

(1) The value 512 is for 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
%SCH	16	16	16	16
%RTC	2	2	2	2
PID	14	14	14	14
Drive objects				
%DRV	16	16	16	16
Communication objects				
%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
User-defined function and user-defined function block objects				
%RET0	1 per user-defined function			
%PARAM	5 per user-defined function and user-defined function block			
%VAR	10 per user-defined function and user-defined function block			
(1) The value 512 is for 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
Motion/Single-axis			
%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			
Motion/Motion Task			
%MC_MotionTask_PTO	0	2	4
Administrative			
%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:

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 timers by specifying the scan period 1...150 ms (default 100 ms) in the periodic mode.

Periodic tasks are triggered by software timers, so are configured by specifying the scan period 1...255 ms (default 255 ms) in the periodic mode.

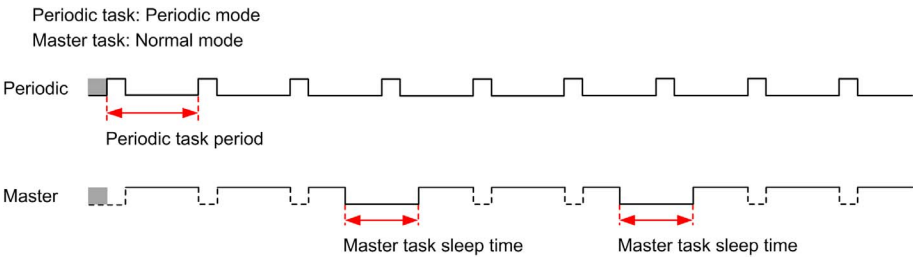
Event tasks are triggered by the physical inputs or the HSC function blocks. These events are associated with embedded digital inputs (%I0.2...%I0.5) (rising, falling or both edges) or with the high speed counters (when the count reaches the high speed counter threshold). You can configure up to two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

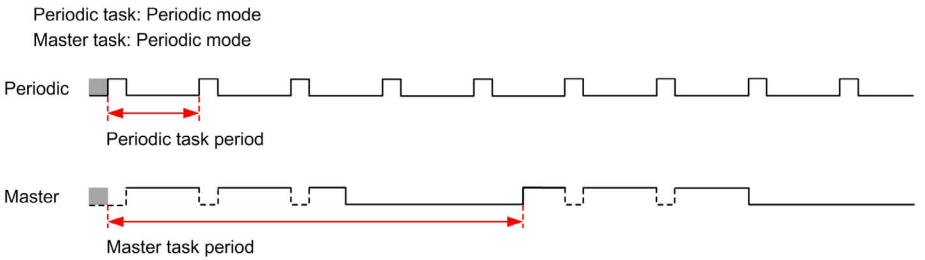
This figure presents the relationship between master tasks and periodic tasks when the master task is in freewheeling mode:



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:

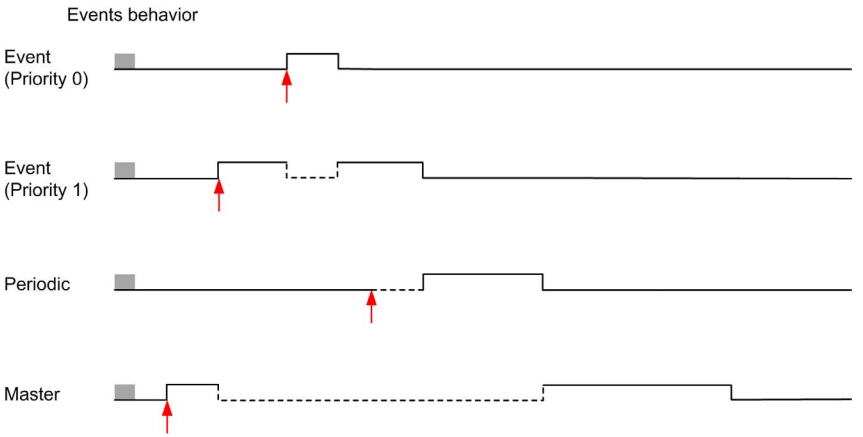


If the processor goes to the `HALTED` state when the master task is configured in freewheeling mode, verify whether the periodic task scan delay time is significant in comparison to the periodic task period. If so, try:

- 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	1...150 ms		
Periodic	Periodic	Software timer	1...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.

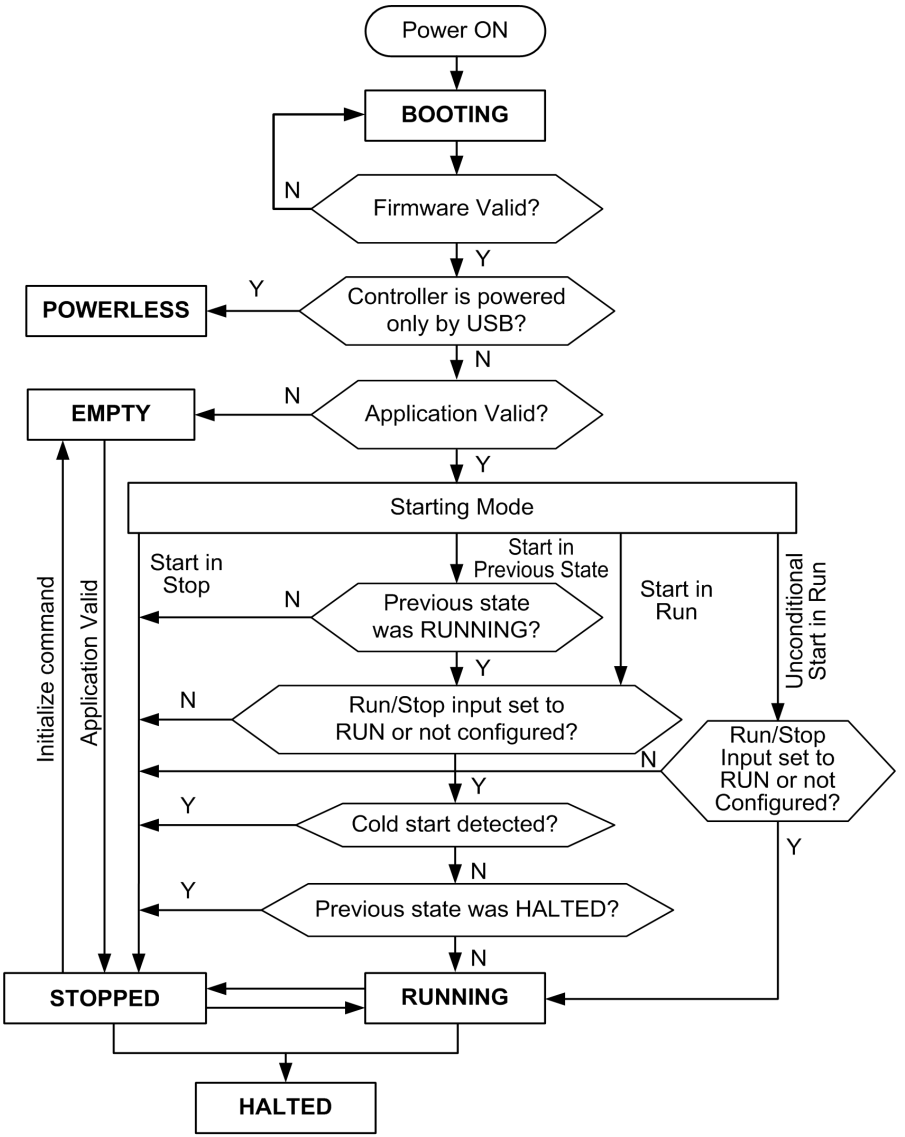
What Is in This Section?

This section contains the following topics:

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.

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

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 *(see page 65)*.	Restricted	No	On	Off	On
EMPTY	This state indicates that there is not a valid application. It is possible to login with SoMachine Basic (*download/animation table*). Inputs are forced to 0. Outputs are set to initialization values *(see page 65)*.	Yes	No	On	Off	1 flash

Controller State	Description	Communication	Application Execution	LED		
				PWR	RUN	ERR
STOPPED	This state indicates that the logic controller has a valid application which is stopped. Inputs are read. Outputs are set to fallback values *(see page 66)*, or forced values *(see page 67)* 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 (animation table, output forcing *(see page 67)*). Status alarm output is set to 1.	Yes	Yes	On	On	Off
HALTED	This state indicates that the application is stopped because an application or system watchdog timeout error has been detected. *(see page 212)* Objects retain their values, allowing analysis of the cause of the detected error. The tasks are stopped at the last instruction. The communication capabilities are the same as in STOPPED state. Inputs are not read, and keep their last values. Outputs are set to fallback values *(see page 66)*. Status alarm output is set to 0.	Yes	No	On	Flashing	On

Controller State	Description	Communication	Application Execution	LED		
				PWR	RUN	ERR
POWERLESS	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). To change the state of the logic controller, connect the main power so that the logic controller boots and reloads the installed components. It is possible to login with SoMachine Basic (*download/upload/animaition table*). Inputs are forced to 0. Outputs are set to initialization values *(see page 65)*.	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 55)*.

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.

Optionally, select the **Reset Memories** option to reset to 0 (default choice) or retain the present value of all memory words and bits on application download *(see SoMachine Basic, Operating Guide)*.

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 *(see Modicon M221 Logic Controller, Hardware Guide)* switch on front face:
  - It commands a transition to RUNNING state on rising edge.
- Run/Stop *(see Modicon M221 Logic Controller, Hardware Guide)* input:
  - The input must be configured in the application (Configuring Digital Inputs *(see page 90)*).
  - It commands a transition to RUNNING state on rising edge.
- SoMachine Basic online button:
  - Select the **Run Controller** command.
- Application starting mode *(see SoMachine Basic, Operating Guide)* 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 *(see Modicon M221 Logic Controller, Hardware Guide)* switch on front face:
  - It forces a transition to STOPPED state on low level.
- Run/Stop *(see Modicon M221 Logic Controller, Hardware Guide)* input:
  - The input must be configured in the application (Configuring Digital Inputs *(see page 90)*).
  - It forces a transition to STOPPED state on low level.
- SoMachine Basic online button:
  - Select the **Stop Controller** command.
- Application starting mode *(see SoMachine Basic, Operating Guide)* 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) *(see SoMachine Basic, Operating Guide)*
- System Watchdog timeout (system overrun, over 80% of the processing capacity is used) *(see page 52)*

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 interruption 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 missing or depleted battery.

These automatically saved persistent variables are reinitialized:

- After each new download, if the **Reset Memories** checkbox is selected in download settings *(see SoMachine Basic, Operating Guide)*.
- Following an initialization command.
- On system bit %S0 activation (refer to System Bits *(see page 242)*).

Save by User Request

You can save memory words in the non-volatile memory or in the SD card. To perform the save operation:

1. Select the destination with %S90 (refer to System Bits *(see page 242)*):
   - Set to 0: non-volatile memory (default)
   - Set to 1: SD card
2. Set the number of memory words to be saved in the system word %SW148 (refer to System Words *(see page 254)*).
3. Set the system bit %S93 to 1 (refer to System Bits *(see page 242)*).

When the save operation is finished:

- The system bit %S93 is reset to 0.
- The system bit %S92 is set to 1, indicating that memory words have been successfully saved in non-volatile memory (%S90 set to 0).
- The system word %SW147 indicates the SD card operation result (%S90 set to 1).

NOTE: You can initiate a memory save while the logic controller is in the RUNNING state. However, depending on the number of memory variables you specify, the save operation may not be accomplished within a single logic scan cycle. As a consequence, the memory values may not necessarily be consistent because the value of the memory variables can change from one scan to another. If you wish to have a consistent set of values for the variables, consider first putting the logic controller into the STOPPED state.

Restore by User Request

You can restore the previously saved memory words. To perform the restore operation:

1. Set the system bit %S92 to 1.
   The non-volatile memory operation has no effect if %S92 is 0 (no values were previously saved).
2. Select the source with %S90 (refer to System Bits *(see page 242)*):
   - Set to 0: non-volatile memory (default)
   - Set to 1: SD card
3. To restore from the non-volatile memory, set the number of memory words in the system word %SW148 (refer to System Words *(see page 254)*). When restoring from SD card, the complete Memory Variables.csv file is processed.
4. Set the system bit %S94 to 1 (refer to System Bits *(see page 242)*).

When the restore operation is finished:

- The system bit %S94 is reset to 0 by the system.
- The system word %SW148 is updated with the number of objects restored (for example if you specify 100 words to restore and only 50 had previously been saved, the value of %SW148 will be 50).
- The system word %SW147 indicates the SD card operation result (%S90 set to 1).

Delete by User Request

You can delete the previously saved memory words on the non-volatile memory. To perform the delete operation:

- Set the system bit %S91 to 1 (refer to System Bits *(see page 242)*).
- When the delete operation is finished, the system bits %S91 and %S92 and the system word %SW148 are reset to 0 by the logic controller.

This operation does not erase the variables in RAM memory.

NOTE: It is not possible to delete only selected variables; the entire set of saved variables is deleted (meaning %SW148 has no impact on the erase operation, the erase operation is carried out regardless of the value of %SW148).

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 behavior *(see SoMachine Basic, Operating Guide)*
  - Fallback values
  - Maintain 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 source transistor output: 0 Vdc
  - Fast sink transistor output: 24 Vdc
  - Regular source transistor output: 0 Vdc
  - Regular sink transistor output: 24 Vdc
  - Relay output: Open
- For expansion module outputs:
  - Regular source transistor output: 0 Vdc
  - Regular sink transistor output: 24 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`), network objects (`%QWE` and `%QWM`), and Modbus Serial IOScanner input objects (`%IN` and `%IWN`) are set to 0. Output objects (`%Q` and `%QW`), network objects (`%IWE` and `%IWM`), and Modbus Serial IOScanner output objects (`%QN` and `%QWN`) are set according to the selected fallback behavior.

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 **Fallback values** is selected, on a fallback occurrence, output values take the values configured in **Fallback value**.
- When **Maintain values** 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 **Fallback values** is selected, the outputs take the values configured in **Fallback value**.
- If **Maintain values** is selected, the outputs keep their values.

Special cases:

- Alarm output, PTO, and FREQGEN: The fallback is never applied. Their fallback values are forced to 0.
- PLS, PWM) and reflex outputs:
  - If **Fallback values** is 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 their fallback 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. However, using system bit %S9 to apply fallback values does not modify the values of 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):
    - Source output: 0 Vdc
    - Sink output: 24 Vdc

- For expansion module outputs:
  - Regular transistor output: according to fallback setting
  - Relay output: according to fallback setting

NOTE: An exception to the application of fallback values is in the case of an I/O expansion bus error. For more information, refer to I/O Configuration General Description *(see page 124)*.

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 can force the value of an output while your controller is connected to SoMachine Basic or with a TMH2GDB Remote Graphic Display *(see Modicon TMH2GDB, Remote Graphic Display, User Guide)*.

To do so, either use the **Force** command in an animation table, or force the value using the F0 or F1 buttons in the Ladder editor.

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

The forcing is not released by any online change nor logout of SoMachine Basic.

The forcing is automatically released by Cold Start *(see page 62)* and Download application *(see page 60)* command.

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

⚠ 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.

NOTE: The output rearming does not apply to sink outputs.

⚠ 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 in 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:

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 62)*
- after a Cold Start *(see page 62)*
- after a reboot *(see page 60)*
- after an application download *(see page 60)*
- after an Ethernet reconfiguration caused by an Ethernet cable reconnection (exclusively for the Ethernet part of the post configuration file *(see page 137)*)

For further details on controller states and transitions, refer to Controller States and Behaviors *(see page 54)*.

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 210)*.

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 136)*).
- 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 210)*.

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 211)*.

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 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:

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

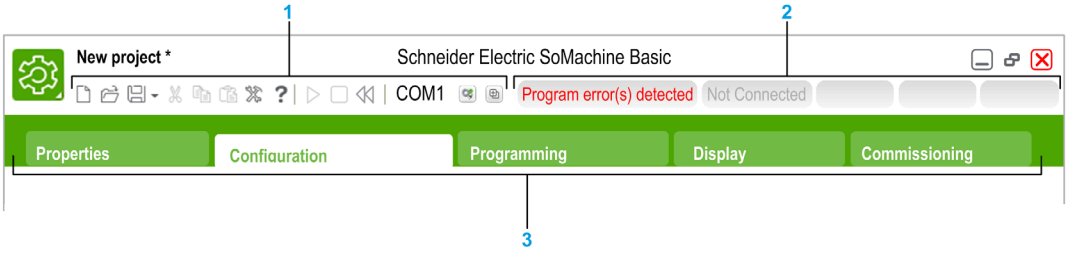
Once you have selected a project to work with, SoMachine Basic displays the main window.

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 Tabs

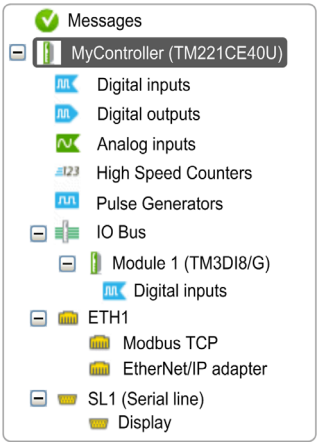
Item	Description
Toolbar	Provides easy access to commonly used functions. For more information, refer to the Toolbar *(see SoMachine Basic, Operating Guide)*.
Status bar	Displays status and information messages on the system status. For more information, refer to the Status bar *(see SoMachine Basic, Operating Guide)*.
Tabs	To develop an application, work your way through the module tabs from left to right: • **Properties** Set up the project properties. • **Configuration** Replicate and configure the hardware configuration of the logic controller and associated expansion modules. • **Programming** Develop the program in one of the supported programming languages. • **Display** Build an operator interface for a Remote Graphic Display device. Refer to the TMH2GDB Remote Graphic Display User Guide for details. • **Commissioning** Manage the connection between SoMachine Basic and the logic controller, upload/download applications, test, and commission the application.

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
Digital inputs	Use to configure the embedded digital inputs of the logic controller.
Digital outputs	Use to configure the embedded digital outputs of the logic controller.
Analog inputs	Use to configure the embedded analog inputs of the logic controller.
High Speed Counters	Use to configure the embedded high speed counting functions (HSC).
Pulse Generators	Use to configure the embedded pulse generator functions (PLS/PWM/PTO/FREQGEN).
IO Bus	Use to configure the expansion modules and cartridges connected to the logic controller.
ETH1	Use to configure the embedded Ethernet communications.
Modbus TCP	Use to configure the Modbus TCP protocol for Ethernet communications.
EtherNet/IP adapter	Use to configure the EtherNet/IP adapter for Ethernet communications.
SLn **(Serial line)**	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.

5 V	24 V
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 *(see SoMachine Basic, Operating Guide)* 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

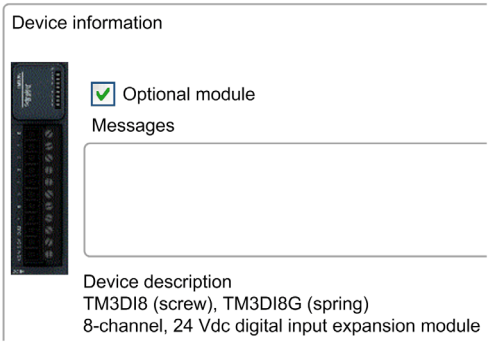
UNINTENDED EQUIPMENT OPERATION

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	In the **Device information** area, select the **Optional module** check box: 

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 **Device information** area, select the **Optional module** 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 **Optional module** feature).

For more information, refer to System Words *(see page 254)*.

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.

If you declare two modules with the same internal ID code next to each other in the configuration and both are declared as optional, a message appears at the bottom of the **Configuration** window. There must be at least one non-optional module between two optional modules.

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

Modules sharing the same internal ID code
TM2DDI16DT, TM2DDI16DK
TM2DRA16RT, TM2DDO16UK, TM2DDO16TK
TM2DDI8DT, TM2DAI8DT
TM2DRA8RT, TM2DDO8UT, TM2DDO8TT
TM2DDO32TK, TM2DDO32UK

Modules sharing the same internal ID code
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	X	X	X	X	–	–	X
TM221C••U TM221CE••U	X	X	X	X	X	–	X
TM221ME16R• TM221CE••R	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 90)*
- Digital outputs, refer to Configuring Digital Outputs *(see page 94)*
- Analog inputs, refer to Configuring Analog Inputs *(see page 96)*
- High speed counters, refer to Configuring High Speed Counters *(see page 99)*
- Pulse generators, refer to Configuring Pulse Generators *(see page 111)*
- Ethernet, refer to Configuring Ethernet *(see page 136)*
- Serial lines, refer to Configuring Serial Line *(see page 176)*.

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 54)* 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 **Start → Programs → Schneider Electric → SoMachine Basic → SoMachine Basic Firmware Update** or run the *ExecLoaderWizard.exe* from *SoMachine Basic installation folder\Execloader* 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 28)*

What Is in This Chapter?

This chapter contains the following sections:

Section 4.1
Digital Input Configuration

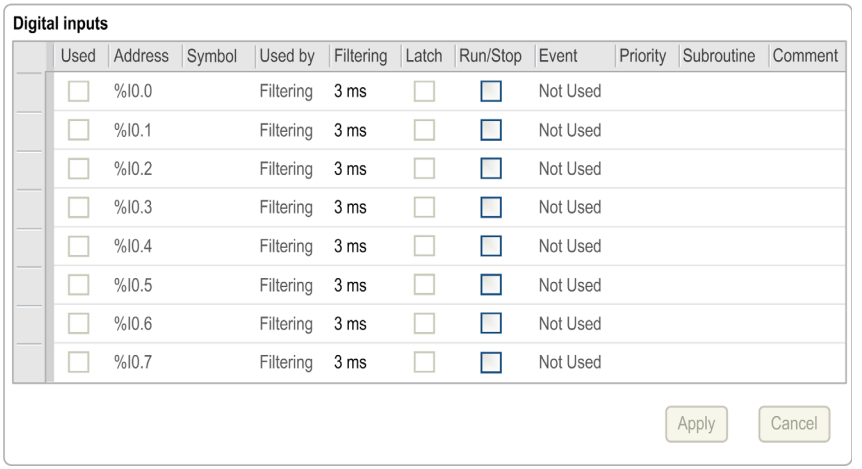
Configuring Digital Inputs

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 99)* while other inputs can be configured as event sources.

Digital Inputs Configuration

This table describes how to configure the digital inputs:

Step	Action
1	Click the **Digital inputs** node in the hardware tree to display the digital input properties. This figure shows the properties of the digital inputs in the editor area: 
2	Edit the properties to configure the digital inputs. For detailed information on the digital input configuration parameters, refer to the table below.

This table describes each parameter of the digital input configuration:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the input channel is being used in a program or not.
Address	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.
Symbol	Yes	–	–	Allows you to specify a symbol to associate with the digital input object. Double-click in the **Symbol** column, type the name of the symbol and press **Enter**.
Used by	No	*any*	**Filtering**	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 **User logic**. The possible values in this field are: • **User logic** • **Filtering** • **Latch** • **Run/Stop** • **Event** • **%HSCx** where x is the high speed counter instance on the controller • **%FC**y where y is the fast counter instance on the controller If an input is being used by more than one operation, all values, separated by commas, are displayed in this field.
Filtering	Yes	**No Filter** **3 ms** **12 ms**	**3 ms**	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: • **Latch** • **Event**

Parameter	Editable	Value	Default Value	Description
Latch	Yes	True/False	False	Allows you to enable or disable latching for the inputs configured as events (`%I0.2...%I0.5`). By default, this option is disabled due to default value of **Filtering**. Set the **Filtering** to **No Filter** to enable the **Latch** option. Latching enables pulses with a duration shorter than the controller scan time to be memorized. 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. If you enable **Latch** for an input, you cannot configure that input for: • **Filtering** • **Run/Stop** • **Event**
Run/Stop	Yes	True/False	False	Allows you to configure 1 digital input as an additional Run/Stop switch. 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). If you enable **Run/Stop** for an input, you cannot configure that input for: • **Latch** • **Event**
Event	Yes	**Not Used** **Falling Edge** **Rising Edge** **Both edges**	**Not Used**	Allows you to select an event that triggers the inputs `%I0.2...%I0.5`. By default, this option is disabled due to the default value of **Filtering**. Set **Filtering** to **No Filter** to enable the **Event** option. When you select an event from the drop-down list (other than **Not Used**): • The **Priority** parameter is enabled to allow you to set the priority of the event. • An event task is created and displayed *(see SoMachine Basic, Operating Guide)* in the **Configuration** tab.
Priority	Yes	0...7	7	Allows you to set the priority of the triggering event for the inputs `%I0.2...%I0.5`. You can set the priority of each event using the **Priority** parameter that is editable only for the inputs configured as event. Assign each configured event a different priority: if 2 events have same priority, a detected error message appears in the window.

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

Additional configuration details are displayed in the **Programming** tab. For more information, refer to Digital Inputs (%I) *(see page 220)*.

Section 4.2
Digital Output Configuration

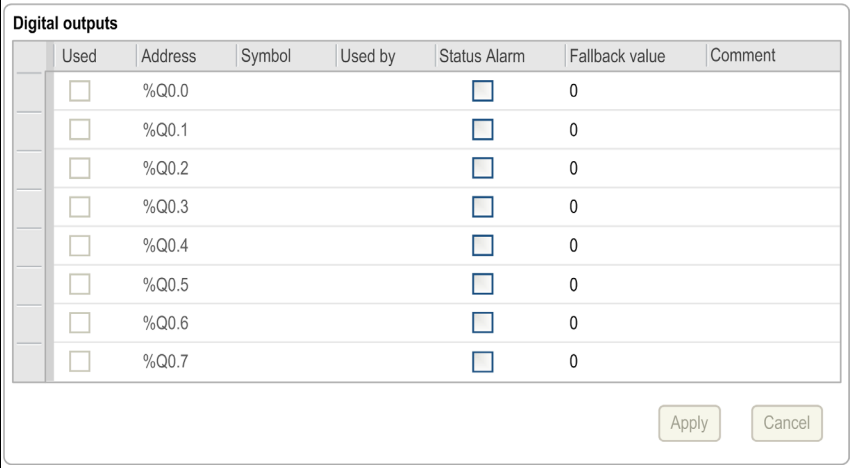
Configuring Digital Outputs

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 111)*.

Digital Outputs Configuration

This table describes how to configure the digital outputs:

Step	Action
1	Click the **Digital outputs** node in the hardware tree to display the digital output properties. This figure shows the properties of the digital outputs in the editor area: 
2	Edit the properties to configure the digital outputs. For detailed information on the digital output configuration parameters, refer to the table below.

This table describes each parameter of the digital output configuration:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the output channel is being used in a program or not.
Address	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.
Symbol	Yes	–	–	Allows you to specify a symbol to associate with the digital output object. Double-click in the **Symbol** column, type the name of the symbol and press **Enter**.
Used by	No	*any*	*empty*	Displays the name of the component that uses the output channel. For example, if the output channel is used as status alarm, it displays **Alarm**.
Status Alarm	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 `RUNNING`, and 0 in all other state
Fallback value	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 `STOPPED` or an exception state. The default value is 0. If **Maintain values** fallback mode is configured, the output retains its current value when the logic controller enters the `STOPPED` or an exception state. This field is disabled for the output configured as **Status Alarm**.
Comment	Yes	–	–	Allows you to specify a comment to associate with the digital output object. Double-click in the **Comment** column, type the comment and press **Enter**.

Additional configuration details are displayed in the **Programming** tab. For more information, refer to Digital Outputs (%Q) *(see page 221)*.

Section 4.3
Analog Input Configuration

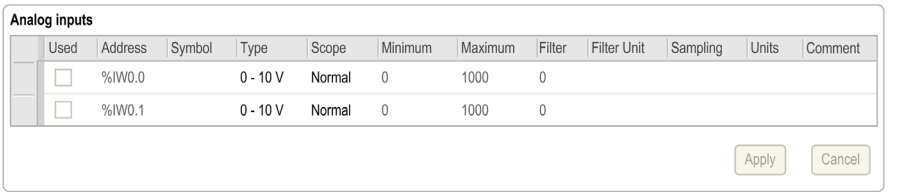
Configuring Analog Inputs

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	Click the **Analog inputs** node in the hardware tree to display the analog input properties. This figure shows the properties of the analog inputs in the editor area: 
2	Edit the properties to configure the analog inputs. For detailed information on the analog input configuration parameters, refer to the table below.

This table describes each parameter of the analog input configuration:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the input channel is being used in a program or not.
Address	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.
Symbol	Yes	–	–	Allows you to specify a symbol to associate with the analog input object. Double-click in the **Symbol** column, type the name of the symbol and press **Enter**.

Parameter	Editable	Value	Default Value	Description
Type	No	**0 - 10 V**	**0 - 10 V**	Indicates the channel mode. For example, **0 - 10 V** refers to the channel that can be used for an electrical input of voltage type in the range 0...10 V.
Scope	No	**Normal**	**Normal**	Indicates the range of values for a channel.
Minimum	No	0	0	Indicates the lower measurement limit.
Maximum	No	1000	1000	Indicates the upper measurement limit.
Filter	No	0	0	Indicates the filtering value. Multiply by the **Filter Unit** value to obtain the filtering time.
Filter Unit	No	100 ms	*empty*	Specifies the unit of time for the filtering value.
Sampling	No	–	*empty*	–
Units	No	*any*	*empty*	Indicates the unit of the analog input.
Comment	Yes	–	–	Allows you to specify a comment to associate with the analog input object. Double-click in the **Comment** column, type the comment and press **Enter**.

Additional configuration details are displayed in the **Programming** tab. For more information, refer to Analog Inputs (%IW) *(see page 222)*.

Section 4.4
High Speed Counter Configuration

What Is in This Section?

This section contains the following topics:

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 *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)* 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

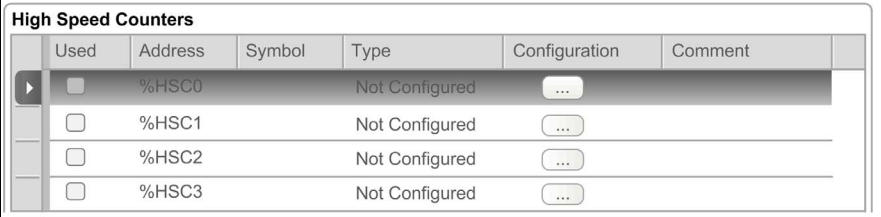
	Main Inputs		Auxiliary Inputs		Reflex Outputs	
%HSC0	**%I0.0**	**%I0.1**	**%I0.2**	**%I0.3**	**%Q0.2**	**%Q0.3**
%HSC1	**%I0.6**	**%I0.7**	**%I0.5**	**%I0.4**	**%Q0.4**	**%Q0.5**
Quadrature X1	Pulse input Phase A	Pulse input Phase B	Preset input	Catch input	Reflex output 0	Reflex output 1
Quadrature 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	Click the **High Speed Counters** node in the hardware tree. **Result:** The **High Speed Counters** list is displayed: 
2	Click **...** in under **Configuration** to select the type of high speed counter to assign and to display the **High Speed Counter Assistant** window. For detailed information on the high speed counter, refer to the table below.

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

Parameter	Editable	Value	Default Value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the high speed counter is being used in a program or not.
Address	No	`%HSCi`		Indicates the address of the high speed counter, where `i` is the object number.
Symbol	Yes	–	–	Allows you to specify a symbol to associate with the high speed counter object. Double-click in the **Symbol** column to edit the field.

Type	No	`Not Configured`
		`Single Phase`
		`Dual Phase`
		`Frequency Meter`
			`Not Configured`	Indicates the counter operational mode.

Configuration	Yes	**[...]** (Button)	Disabled	Allows you to configure the high speed counter parameters using the **High Speed Counter Assistant** window.
Comment	Yes	–	–	Allows you to specify a comment to associate with the high speed counter object. Double-click in the **Comment** column to edit the field.

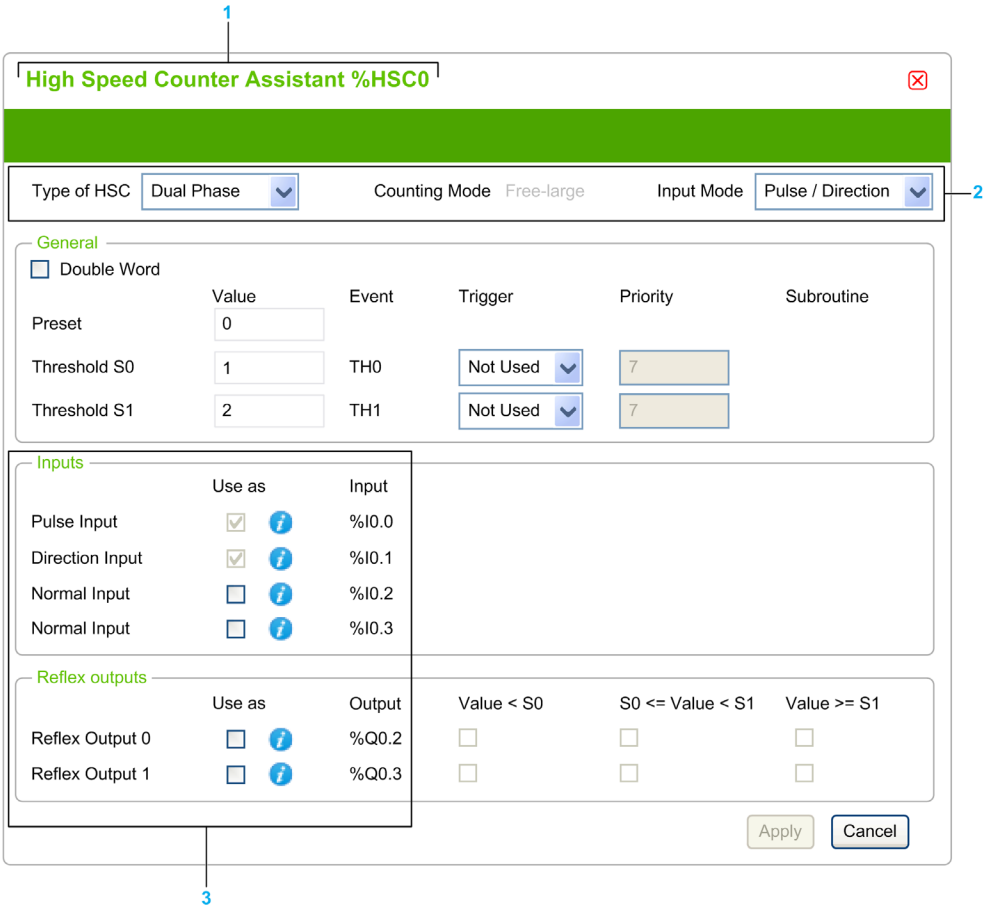
For details on the configuration of the `Dual Phase [Pulse / Direction]`, `Dual Phase [Quadrature X1]`, `Dual Phase [Quadrature X2]`, `Dual Phase [Quadrature X4]`, and `Single Phase`, refer to Configuring Dual Phase and Single Phase Counters *(see page 103)*.

For details on the configuration of the `Frequency Meter`, refer to Configuring Frequency Meter *(see page 108)*.

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.

Item	Description
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 *(see page 100)*.

Common Parameters

This table describes parameters common to all counter types:

Parameter	Editable	Value	Default Value	Description

Type of HSC	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 *(see page 100)*.

Counting Mode	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 *(see page 100)*.

Input Mode	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 *(see page 100)*.

Double Word	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).
Preset	Yes	0...65535 (Word)	0 (Word)	Allows you to specify the preset value for counting functions.
		0...4294967295 (Double Word)	0 (Double Word)	
Threshold S0	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)	

Parameter	Editable	Value	Default Value	Description
Threshold S1	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)	
Trigger	Yes	**Not Used** **Falling Edge** **Rising Edge** **Both edges**	**Not Used**	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 **Priority** parameter editable.
Priority	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.
Subroutine	No	*any*	*empty*	Displays the subroutine associated with an input configured as an event (for both threshold `TH0` and `TH1`).
Normal Input	Yes	TRUE/FALSE	FALSE	Configurable as **Preset Input** by selecting the **Use as** check box, only on `%HSC0` and `%HSC1`, respectively `%I0.2` and `%I0.5`.
Normal Input	Yes	TRUE/FALSE	FALSE	Configurable as **Catch Input** by selecting the **Use as** check box, only on `%HSC0` and `%HSC1`, respectively `%I0.3` and `%I0.4`.
Reflex Output 0	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`.
Reflex Output 1	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`.
Value < S0	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`.

Parameter	Editable	Value	Default Value	Description
S0 <= Value < S1	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`.
Value >= S1	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`. • TRUE = down counting • FALSE = up counting

`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

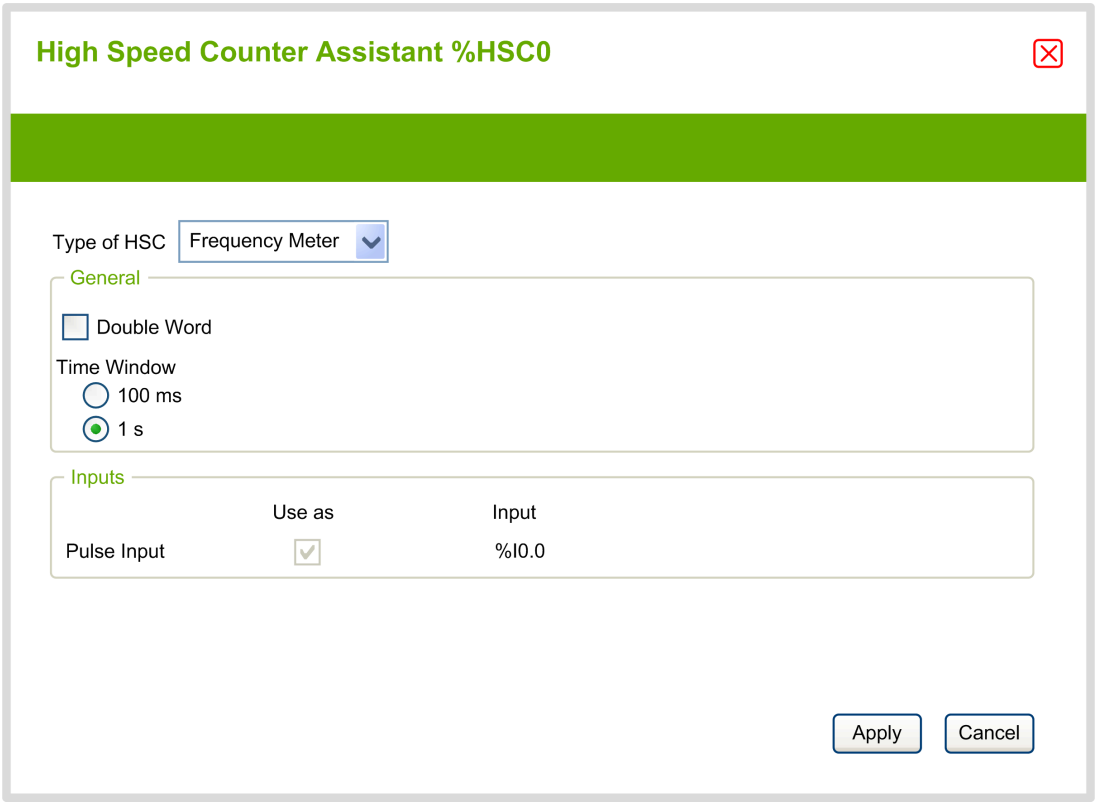
This table describes a parameter specific to `Single Phase`:

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: • `%I0.0` for `%HSC0` • `%I0.6` for `%HSC1` • `%I0.1` for `%HSC2` • `%I0.7` for `%HSC3`

Configuring Frequency Meter

High Speed Counter Assistant

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



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

Type of HSC	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 *(see page 100)*.

Double Word	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).
Time Window	Yes	100 ms 1 s	1 s	Allows you to select the time base to measure the frequency between 100 Hz and 100 kHz.
Pulse Input	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:

Configuring Pulse Generators

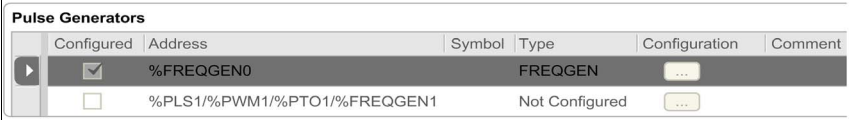
Introduction

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 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	Click the **Pulse Generators** node in the hardware tree to display the pulse generator properties. This figure presents the properties of the pulse generators in the editor area: 
2	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.

This table describes the parameters of the pulse generator:

Parameter	Editable	Value	Default Value	Description
Used	No	True/False	False	Indicates whether the pulse generated output is being used in a program or not.
Address	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.
Symbol	Yes	–	–	Allows you to specify a symbol to associate with the pulse generator object. Double-click in the **Symbol** column to edit the field.
Type	No	**Not Configured** **PLS** **PWM** **PTO** **FREQGEN**	**Not Configured**	Displays the type of the pulse generator used for the output channel.

Parameter	Editable	Value	Default Value	Description
Configuration	Yes	**[...]** (Button)	Enabled	Allows you to configure the pulse generator using the **Pulse Generator Assistant** window.
Comment	Yes	–	–	Allows you to specify a comment to associate with the pulse generator object. Double-click in the **Comment** column to edit the field.

PLS Configuration

Refer to Configuring Pulse (%PLS) *(see page 113)*.

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

PWM Configuration

Refer to Configuring Pulse Width Modulation (%PWM) *(see page 116)*.

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) *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*.

PTO Configuration

Refer to Configuring Pulse Train Output (%PTO) *(see page 118)*

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) *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*.

Frequency Generator Configuration

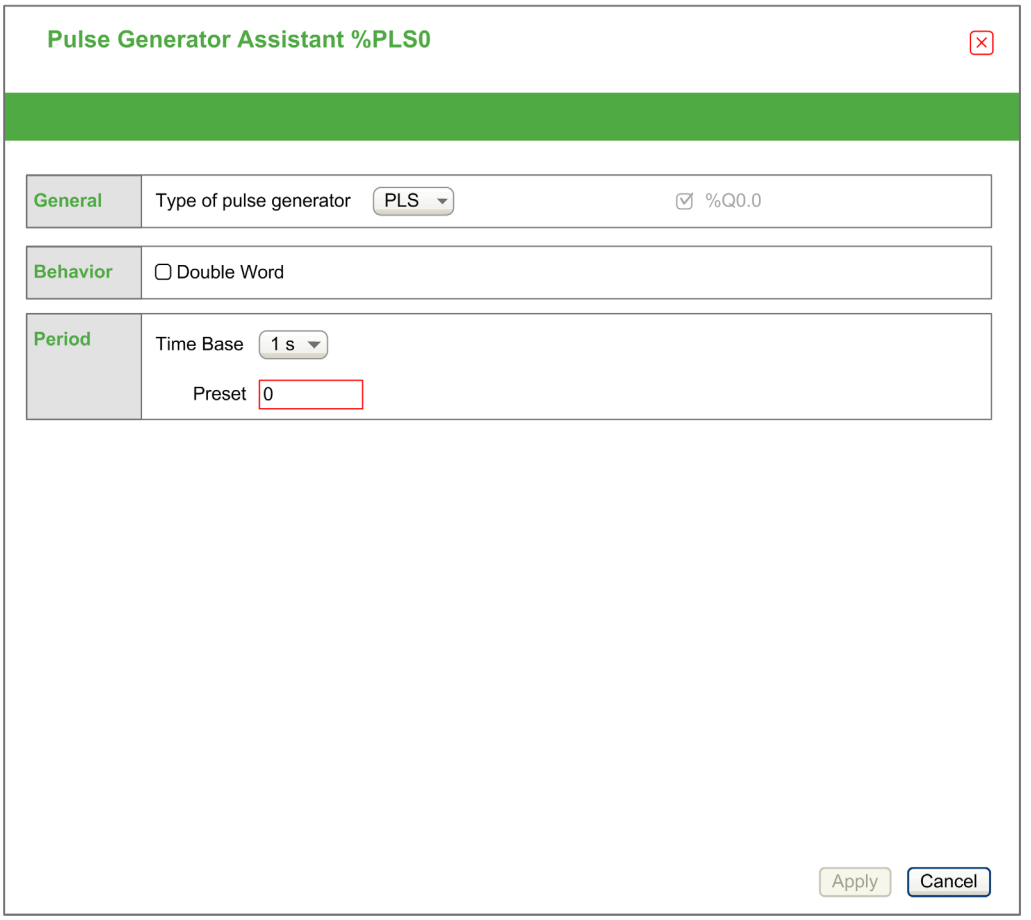
Refer to Configuring Frequency Generator (%FREQGEN) *(see page 118)*

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

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**:



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

Parameter	Value	Default Value	Description
Type of pulse generator	**Not Configured** **PLS** **PWM** **PTO** **FREQGEN**	**PLS**	Allows you to choose the type of pulse generator and configure the output properties. Select: ● **PLS** to configure the output channels in `PLS` mode. Refer to Configuring Pulse (%PLS) *(see page 113)*. ● **PWM** to configure the output channels in `PWM` mode. Refer to Configuring Pulse Width Modulation (%PWM) *(see page 116)*. ● **PTO** to configure the output channels in `PTO` mode. Refer to Configuring Pulse Train Output (%PTO) *(see page 118)*. ● **FREQGEN** to configure the output channels in `FREQGEN` mode. Refer to Configuring Frequency Generator (%FREQGEN) *(see page 121)*.
Double Word	True/False	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).
Time Base	**0.1 ms** **1 ms** **10 ms** **1 s**	**1 s**	Allows you to select the time base for the frequency measurement.
Preset	Refer to the table below for complete range of preset values for `PLS` 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

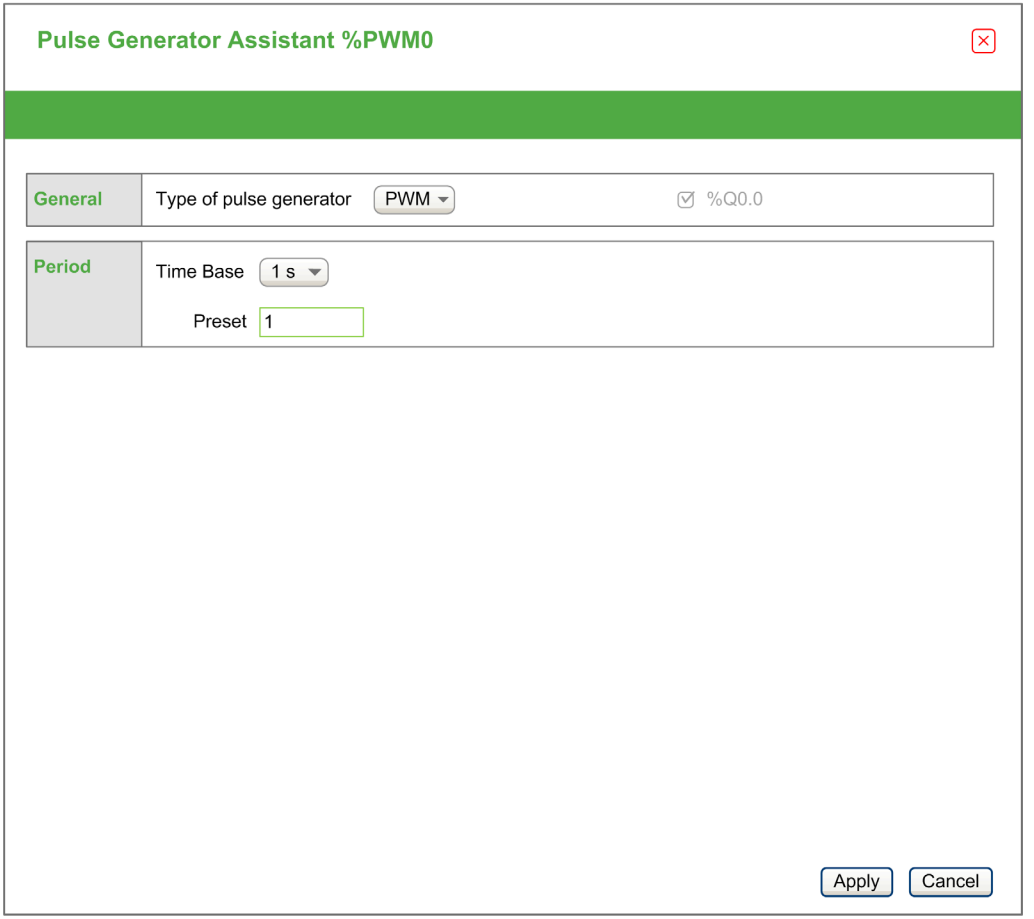
Additional configuration details are displayed in the **Programming** tab.

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

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**:



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

Parameter	Value	Default Value	Description
Type of pulse generator	**Not Configured** **PLS** **PWM** **PTO** **FREQGEN**	**PWM**	Allows you to choose the type of pulse generator and configure the output properties. Select: ● **PLS** to configure the output channels in `PLS` mode. Refer to Configuring Pulse (%PLS) *(see page 113)*. ● **PWM** to configure the output channels in `PWM` mode. Refer to Configuring Pulse Width Modulation (%PWM) *(see page 116)*. ● **PTO** to configure the output channels in `PTO` mode. Refer to Configuring Pulse Train Output (%PTO) *(see page 118)*. ● **FREQGEN** to configure the output channels in `FREQGEN` mode. Refer to Configuring Frequency Generator (%FREQGEN) *(see page 121)*.
Time Base	**0.1 ms** **1 ms** **10 ms** **1 s**	**1 s**	Allows you to select the time base for the frequency measurement.
Preset	Refer to the table below for complete range of preset values for `PWM` type pulse generator.	0	Allows you to specify the preset value for the `PWM` 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

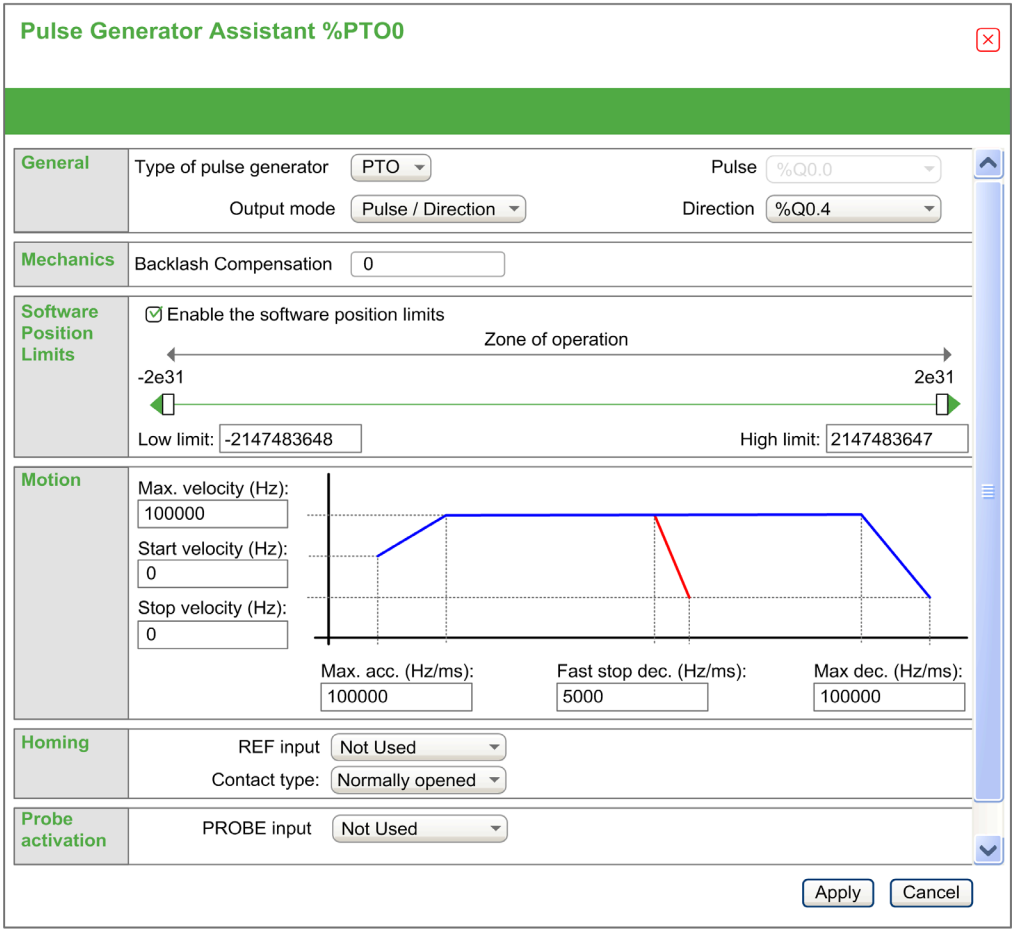
Additional configuration details are displayed in the **Programming** tab.

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) *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*.

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**:



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

Parameter		Value	Default	Description
General	**Type of pulse generator**	**Not Configured** **PLS** **PWM** **PTO** **FREQGEN**	**PTO**	Allows you to choose the type of pulse generator and configure the output properties. Select: • **PLS** to configure the output channels in `PLS` mode. Refer to Configuring Pulse (%PLS) *(see page 113)*. • **PWM** to configure the output channels in `PWM` mode. Refer to Configuring Pulse Width Modulation (%PWM) *(see page 116)*. • **PTO** to configure the output channels in `PTO` mode. Refer to Configuring Pulse Train Output (%PTO) *(see page 118)*. • **FREQGEN** to configure the output channels in `FREQGEN` mode. Refer to Configuring Frequency Generator (%FREQGEN) *(see page 121)*.
	Output mode	**Clock Wise / Counter Clock Wise** **Pulse / Direction**	**Pulse / Direction**	Select the pulse Output mode *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*. **NOTE:** The **Clock Wise / Counter Clock Wise** output mode is only valid for PTO0. This mode disables PTO1.
	Pulse	%Q0.0 for PTO0, %Q0.1 for PTO1	%Q0.0 for PTO0, %Q0.1 for PTO1	When **Pulse / Direction** is selected in **Output mode**, select the output that provides the motor operating speed.
	Direction	**Not used** %Q0.0...16 (depending on controller reference)	%Q0.2	When **Pulse / Direction** is selected in **Output mode**, select the output that provides the motor rotation direction. Set to **Not used** (disabled) if directional output is not required for the application. **NOTE:** The application must be configured with a functional level of at least **Level 5.0** to enable the **Not used** option.
	Clock Wise	%Q0.0	%Q0.0	When **Clock Wise / Counter Clock Wise** is selected in **Output mode**, select the output that provides the signal for forward motor operating speed and direction.
	Counter Clock Wise	%Q0.1	%Q0.1	When **Clock Wise / Counter Clock Wise** is selected in **Output mode**, select the output that provides the signal for reverse motor operating speed and direction.

Parameter		Value	Default	Description
Mechanics	**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 *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*.
Software Position Limits	**Enable the software position limits**	Enabled Disabled	Enabled	Select whether to use the software position limits.
	Low Limit	**-2147483648... 2147483647**	**-2147483648**	Set the software limit position to be detected in the negative direction.
	High Limit	**-2147483648... 2147483647**	**2147483647**	Set the software limit position to be detected in the positive direction.
Motion	**Max. velocity**	0...100000	100000	Set the pulse output maximum velocity (in Hz).
	Start velocity	0...100000	0	Set the pulse output start velocity *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)* (in Hz). 0 if not used.
	Stop velocity	0...100000	0	Set the pulse output stop velocity *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)* (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).
Homing	**REF input**	**Not Used** Input	**Not Used**	Select whether to use the REF input to set the homing position.
	Contact type	**Normally opened** **Normally closed**	**Normally opened**	Select whether the switch contact default state is open or closed. **NOTE:** The input type is only available when **REF input** is selected.
Probe activation	**PROBE input**	**Not Used** Input	**Not Used**	Select whether to use the PROBE input. **NOTE:** Refer to Regular Input Characteristics for details on the physical characteristics of the selected input.

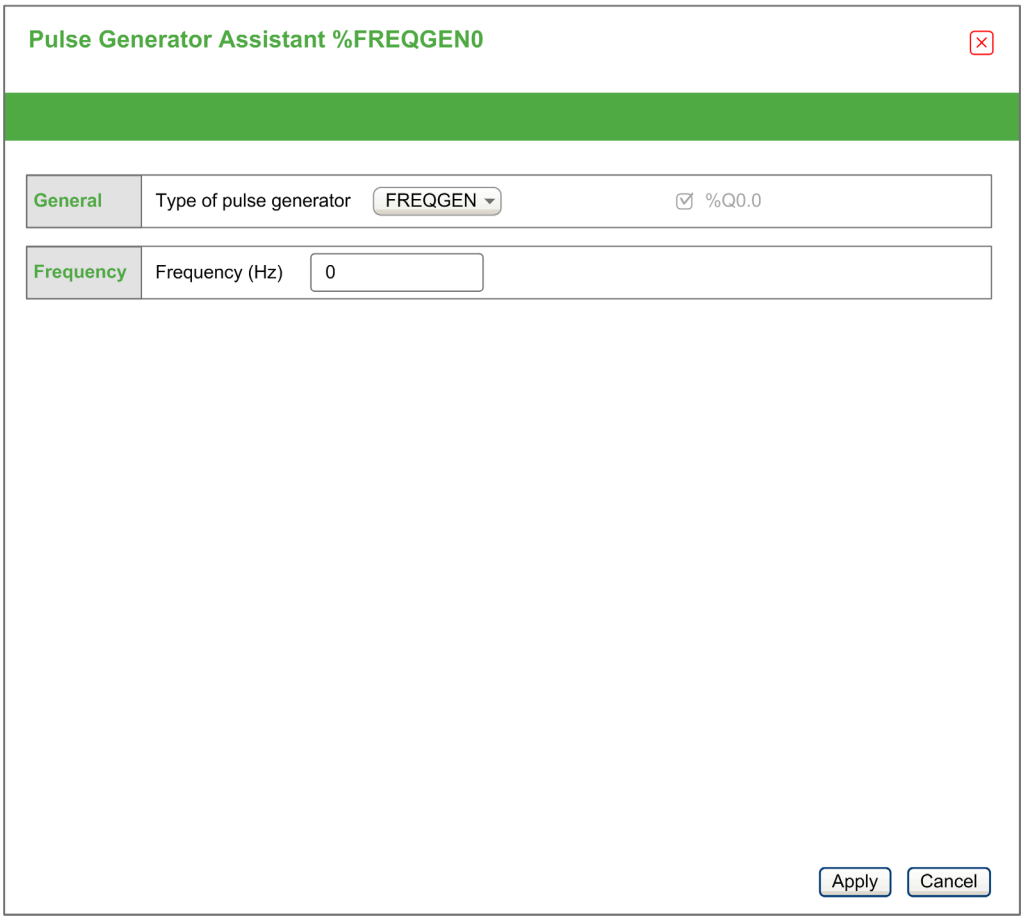
Additional configuration details are displayed in the **Programming** tab.

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) *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*.

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**:



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) *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*.

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:

I/O Configuration General Description

Introduction

In your project, you can add I/O expansion modules to your M221 Logic Controller to increase the number of digital and analog inputs and outputs over those native to the logic controller itself (embedded I/O).

You can add either TM3 or TM2 I/O expansion modules to the logic controller, and further expand the number of I/O via TM3 transmitter and receiver modules to create remote I/O configurations. Special rules apply in all cases when creating local and remote I/O expansions, and when mixing TM2 and TM3 I/O expansion modules (refer to Maximum Hardware Configuration *(see page 128)*).

The I/O expansion bus of the M221 Logic Controller is created when you assemble the I/O expansion modules to the logic controller. I/O expansion modules are considered as external devices in the logic controller architecture and are treated, as such, differently than the embedded I/Os of the logic controller.

I/O Expansion Bus Errors

If the logic controller cannot communicate with one or more I/O expansion modules that is (are) contained in the program configuration and those modules are not configured as optional modules (refer to Optional I/O Expansion Modules *(see page 83)*), the logic controller considers it as an I/O expansion bus error. The unsuccessful communication may be detected during the startup of the logic controller or during runtime, and there may be any number of causes. Causes of communication exception on the I/O expansion bus include, among other things, disconnection of or physically missing I/O modules, electromagnetic radiation beyond published environmental specifications, or otherwise, inoperative modules.

During runtime, if an I/O expansion bus error is detected, the diagnostic information is contained in `%SW118` and `%SW120` system words, and the red LED indicator labeled **ERR** flashes.

Active I/O Expansion Bus Error Handling

System bit %S106 is set to 0 by default to specify the use of active I/O error handling. The application can set this bit to 1 to use passive I/O error handling instead.

By default (system bit %S106 set to 0), when the logic controller detects a TM3 module in bus communication error, it sets the bus to a "bus off" condition whereby the TM3 expansion module outputs are set to 0. A TM3 expansion module is considered to be in bus communication error when an I/O exchange with the expansion module has been unsuccessful for at least two consecutive bus task cycles. When a bus communication error occurs, bit n of %SW120 is set to 1, where n is the expansion module number, and %SW118 bit 14 is set to 0.

Normal I/O expansion bus operation can only be restored after eliminating the source of the error and performing one of the following:

- Power cycle
- New application download
- Application request through a rising edge on bit `%S107`
- With SoMachine Basic by selection of the **Initialize Controller** command

Passive I/O Expansion Bus Error Handling

The application can set system bit %S106 to 1 to use passive I/O error handling. This error handling is provided to afford compatibility with previous firmware versions and previous controllers that the M221 Logic Controller replaces.

When passive I/O error handling is in use, the controller attempts to continue data bus exchanges with the modules during bus communication errors. While the expansion bus error persists, the logic controller attempts to re-establish communication on the bus with incommunicative modules, depending on the type of I/O expansion module, TM3 or TM2:

- For TM3 I/O expansion modules, the value of the I/O channels is maintained (**Maintain values**) for approximately 10 seconds while the logic controller attempts to re-establish communication. If the logic controller cannot re-establish communications within that time, all affected TM3 I/O expansion outputs are set to 0.
- For the TM2 I/O expansion modules that may be part of the configuration, the value of the I/O channels is maintained indefinitely. That is to say, the outputs of the TM2 I/O expansion modules are set to **Maintain values** until either power is cycled on the logic controller system, or you issue an **Initialize Controller** command with SoMachine Basic.

In either case, the logic controller continues to solve logic and the embedded I/O continues to be managed by the application (Managed by application *(see page 65)*) while it attempts to re-establish communication with the incommunicative I/O expansion modules. If the communication is successful, the I/O expansion modules resume to be managed by the application. If communication with the I/O expansion modules is unsuccessful, you must resolve the reason for the unsuccessful communication, and then cycle power on the logic controller system, or issue an **Initialize Controller** command with SoMachine Basic.

Further, if the incommunicative I/O module(s) disturb the communication with unaffected modules, the unaffected modules will also be considered in error and their corresponding bit in `%SW120` will be set to 1. However, with the ongoing data exchanges that characterize the Passive I/O Expansion Bus Error Handling, the unaffected modules will nonetheless apply the data sent, and will not apply the fallback values as for the incommunicative module.

Therefore, you must monitor within your application the state of the bus and the error state of the module(s) on the bus, and to take the appropriate action necessary given your particular application.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Include in your risk assessment the possibility of unsuccessful communication between the logic controller and any I/O expansion modules.
- If the “Maintain values” option deployed during an I/O expansion bus error is incompatible with your application, use alternate means to control your application for such an event.
- Monitor the state of the I/O expansion bus using the dedicated system words and take appropriate actions as determined by your risk assessment.

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

For more information on the actions taken upon start-up of the logic controller when an I/O expansion bus error is detected, refer to Optional I/O Expansion Modules *(see page 83)*.

Restarting the I/O Expansion Bus

When active I/O error handling is being applied, that is, TM3 outputs set to 0 when a bus communication error is detected, the application can request a restart of the I/O expansion bus while the logic controller is still running (without the need for a Cold Start, Warm Start, power cycle, or application download).

System bit %S107 is available to request restarts of the I/O expansion bus.The default value of this bit is 0. The application can set %S107 to 1 to request a restart of the I/O expansion bus. On detection of a rising edge of this bit, the logic controller reconfigures and restarts the I/O expansion bus if all of the following conditions are met:

- %S106 is set to 0 (that is, I/O expansion bus activity is stopped)
- %SW118 bit 14 is set to 0 (I/O expansion bus is in error)
- At least one bit of %SW120 is set to 1 (at least one expansion module is in bus communication error)

If %S107 is set to 1 and any of the above conditions is not met, the logic controller takes no action.

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 important 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), then you must update your application configuration. This is also true for any field bus devices you may have in your installation. Otherwise, there is the potential that the expansion bus or field bus 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 83)*.

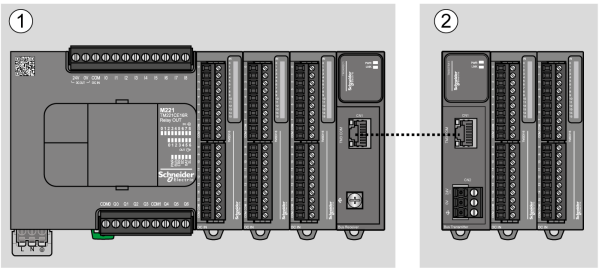
Maximum Hardware Configuration

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

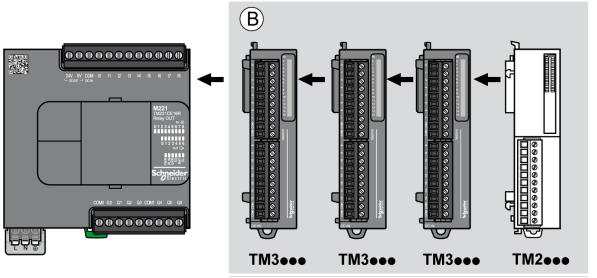
M221 Logic Controller Local Configuration Architecture

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

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

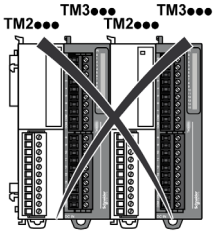
Application requirements determine the architecture of your M221 Logic Controller configuration.

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

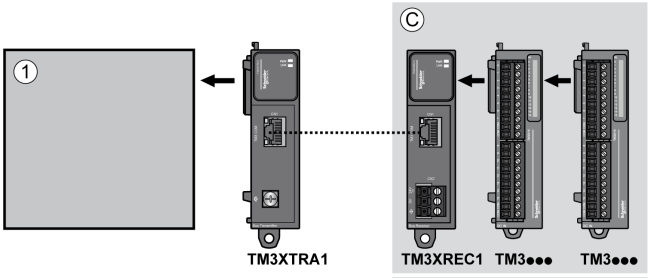
Optimized remote configuration and flexibility are provided by the association of:

- 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• TM221C24• TM221CE24• TM221C40• TM221CE40• TM221M16R• TM221ME16R• TM221M16T• TM221ME16T• TM221M32TK TM221ME32TK	7 TM3 / TM2 expansion modules	Local
TM3XREC1	7 TM3 expansion modules	Remote
NOTE: 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

Reference	IO Bus 5 Vdc	IO Bus 24 Vdc
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 Analog 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 6.1
Ethernet Configuration

What Is in This Section?

This section contains the following topics:

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

NOTE: Schneider Electric adheres to industry best practices in the development and implementation of control systems. This includes a "Defense-in-Depth" approach to secure an Industrial Control System. This approach places the controllers behind one or more firewalls to restrict access to authorized personnel and protocols only.

⚠ WARNING

UNAUTHENTICATED ACCESS AND SUBSEQUENT UNAUTHORIZED MACHINE OPERATION

- Evaluate whether your environment or your machines are connected to your critical infrastructure and, if so, take appropriate steps in terms of prevention, based on Defense-in-Depth, before connecting the automation system to any network.
- Limit the number of devices connected to a network to the minimum necessary.
- Isolate your industrial network from other networks inside your company.
- Protect any network against unintended access by using firewalls, VPN, or other, proven security measures.
- Monitor activities within your systems.
- Prevent subject devices from direct access or direct link by unauthorized parties or unauthenticated actions.
- Prepare a recovery plan including backup of your system and process information.

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

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
Server	8
Client	1

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.

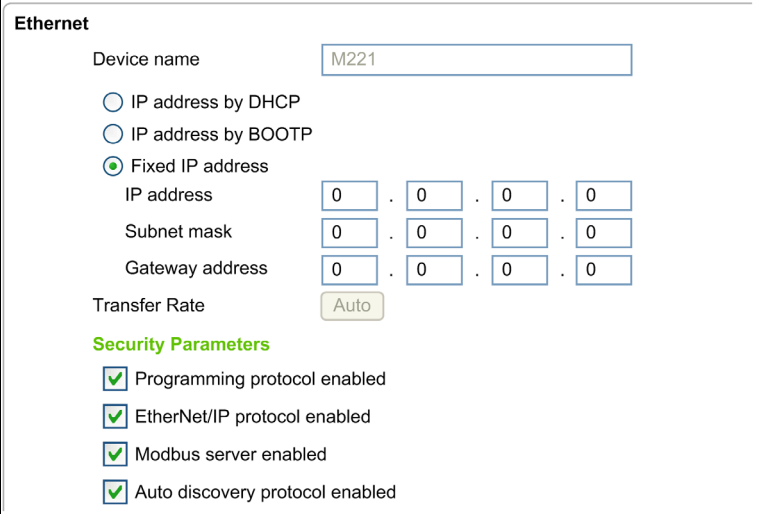
The server connections stay open as long as the logic controller stays in its present operational state (`RUNNING`, `STOPPED`, or `HALTED`).

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	Click the **ETH1** node in the hardware tree to display the Ethernet properties. This figure presents the Ethernet properties in the editor area: 
2	Edit the properties to configure the Ethernet. For detailed information on the Ethernet configuration parameters, refer to the table below.

NOTE: The **Security Parameters** displayed depend on the functional level *(see SoMachine Basic, Operating Guide)* selected for the application.

This table describes each parameter of the Ethernet configuration:

Parameter	Editable	Value	Default Value	Description
Ethernet				
Device name	No	*any*	**M221** (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.
(1) You can select any one option for IP addressing. Selecting any one option disables the other options. **(2)** These options are enabled only if you select the option **Fixed IP Address** for IP addressing. **(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.				

Parameter	Editable	Value	Default Value	Description
IP address by DHCP	Yes(1)	TRUE/FALSE	FALSE	Allows you to obtain the IP address from the DHCP server on the network.
IP address by BOOTP	Yes(1)	TRUE/FALSE	FALSE	Allows you to obtain the IP address from the Boot PROM configuration server on the network.
Fixed IP address	Yes(1)	TRUE/FALSE	TRUE	Allows you to specify the IP address manually for host or network interface identification.
IP address	Yes(2)	*w.x.y.z*(3)	0.0.0.0	Allows you to specify the IP address of the device in the Ethernet network. See Address Classes *(see page 143)* Assigning 0.0.0.0 (the default) 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) = **25** decimal) FF (73 hex) = **155** decimal IP address generated: 10.10.**25.155**. The firmware also generates an IP address from the MAC address if the specified IP address is identified as being a duplicate address on the network. Bit 9 of system word %SW118 is set to 1 (see System Words Description *(see page 255)*) and system word %SW62 is set to 1 (see System Words Description *(see page 255)*) when a duplicated IP address is detected. The MAC address of the logic controller is stored in %SW107-%SW109 (see System Words Description *(see page 255)*).

(1) You can select any one option for IP addressing. Selecting any one option disables the other options.
(2) These options are enabled only if you select the option **Fixed IP Address** for IP addressing.
(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.

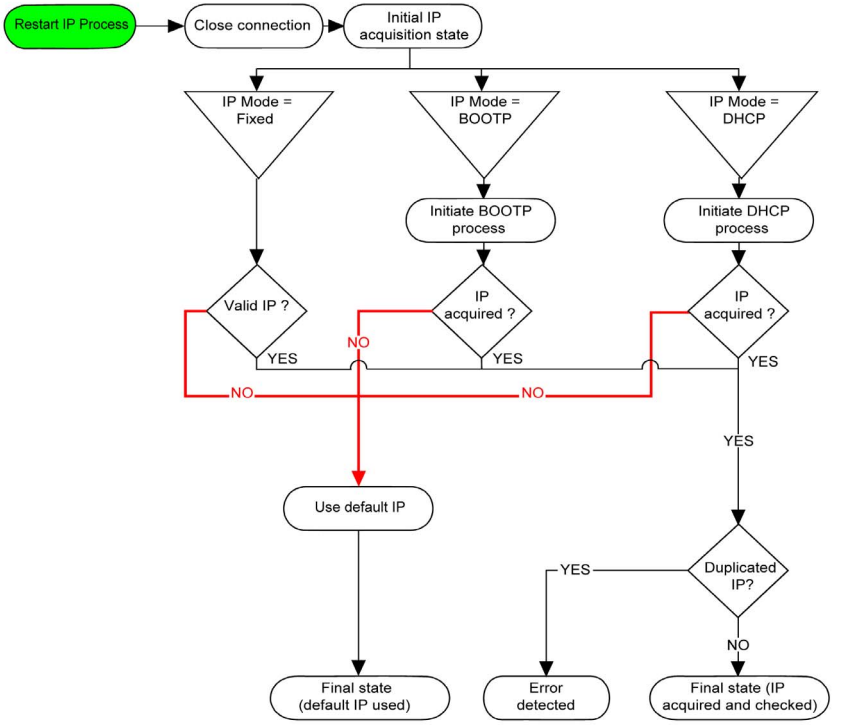
Parameter	Editable	Value	Default Value	Description
Subnet mask	Yes(2)	*w.x.y.z*(3)	0.0.0.0	Allows you to specify the address of the subnetwork to authorize a group of devices for data exchange. It determines which bits in an IP address correspond to the network address and which bits correspond to the subnet portions of the address. See Subnet Mask *(see page 143)*
Gateway address	Yes(2)	*w.x.y.z*(3)	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. See Gateway Address *(see page 143)*
Transfer Rate	No	–	**Auto**	Displays the selected mode for Ethernet speed. Auto stands for “Auto Negotiation".
Security Parameters The security parameters allow you to enable or disable communication protocols and features.				
Programming protocol enabled	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.
EtherNet/IP protocol enabled	Yes	TRUE/FALSE	TRUE	Allows you to enable or disable the EtherNet/IP protocol to connect to a network for data exchange.
Modbus server enabled	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.
Auto discovery protocol enabled	Yes	TRUE/FALSE	TRUE	Allows you to enable or disable the auto discovery protocol to automatically detect devices on supported Ethernet fieldbusses.

(1) You can select any one option for IP addressing. Selecting any one option disables the other options.
(2) These options are enabled only if you select the option **Fixed IP Address** for IP addressing.
(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.

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.

Address Classes

The IP address is linked:

- to a device (the host)
- to the network to which the device is connected

An IP address is always coded using 4 bytes.

The distribution of these bytes between the network address and the device address may vary.This distribution is defined by the address classes.

The different IP address classes are defined in this table:

Address Class	Byte1					Byte 2	Byte 3	Byte 4
Class A	0	Network ID				Host ID		
Class B	1	0	Network ID				Host ID	
Class C	1	1	0	Network ID				Host ID
Class D	1	1	1	0	Multicast Address			
Class E	1	1	1	1	0	Address reserved for subsequent use		

Subnet Mask

The subnet mask is used to address several physical networks with a single network address. The mask is used to separate the subnetwork and the device address in the host ID.

The subnet address is obtained by retaining the bits of the IP address that correspond to the positions of the mask containing 1, and replacing the others with 0.

Conversely, the subnet address of the host device is obtained by retaining the bits of the IP address that correspond to the positions of the mask containing 0, and replacing the others with 1.

Example of a subnet address:

IP address	192 (11000000)	1 (00000001)	17 (00010001)	11 (00001011)
Subnet mask	255 (11111111)	255 (11111111)	240 (11110000)	0 (00000000)
Subnet address	192 (11000000)	1 (00000001)	16 (00010000)	0 (00000000)

NOTE: The device does not communicate on its subnetwork when there is no gateway.

Gateway Address

The gateway allows a message to be routed to a device that is not on the current network.

If there is no gateway, the gateway address is 0.0.0.0.

Configuring Modbus TCP

Introduction

You can configure the Ethernet port for Modbus TCP or Modbus TCP IOScanner as:

- Modbus mapping *(see page 144)*
- Client mode *(see page 146)*

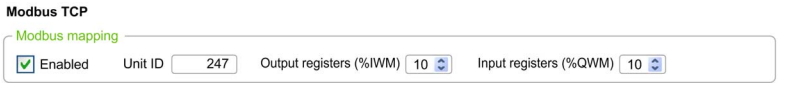
Only one instance of IOScanner can be defined: if you configure it on a serial port, you cannot configure it on an Ethernet port and vice versa. Refer to Configuring Modbus Serial IOScanner *(see page 185)*.

The maximum number of TCP and Serial IOScanner objects is:

- 128, if the **Functional Level** < **6.0**.
- 512, if the **Functional Level** ≥ **6.0**.

Configuring Modbus TCP: Modbus Mapping

This table describes how to configure Modbus mapping:

Step	Action
1	In the **Configuration** window, click **ETH1→ Modbus TCP** to display the Modbus TCP properties. The following illustration shows the properties displayed in the editor area: 
2	Select **Enabled** to edit the properties to configure **Modbus mapping**. **NOTE:** If the **Enabled** button is grayed out, verify that the **Functional Level** of your application (**Programming → Tasks → Behavior** tab) is at least **Level 3.2**.
3	Click **Apply**.

This table describes each parameter of the **Modbus mapping** configuration:

Parameter	Editable(1)	Value	Default Value	Description
Enabled	Yes	TRUE/FALSE	FALSE	Select to enable **Modbus mapping**. **NOTE:** If you deselect the **Enabled** 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 **Security Parameters** for the protocol in the Ethernet properties window *(see page 137)*.
Unit ID	Yes	1...247	-	Specify the unit ID of the local server. Modbus TCP requests originating from a device with the same unit ID are sent to the Modbus mapping table instead of the regular Modbus server.
Output registers (%IWM)	Yes	1...20	10	The number of output registers available. Output registers are used to store the values of Modbus TCP (%IWM) objects *(see page 231)*.
Input registers (%QWM)	Yes	1...20	10	The number of input registers available. Input registers are used to store the values of Modbus TCP (%QWM) objects *(see page 229)*.

(1) Only if the **Modbus server enabled** option is selected in the **Security Parameters** section of the Ethernet properties windows *(see page 142)*.

Modbus TCP Slave Device I/O Mapping Table

When the 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.

If the unit ID selected in the master is not the one configured in the M221 slave (or vice versa), data is read or written to regular Modbus words `%MWx` instead of network objects `%IWMx` and `%QWMx`. No Modbus error is returned.

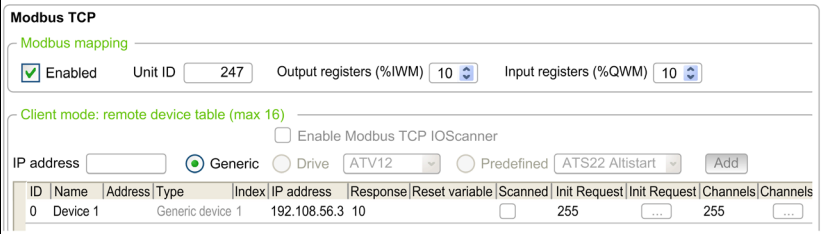
Access to the Modbus TCP slave I/O mapping table (`%IWM`/`%QWM`) is done with the same priority as access to regular Modbus words (`%MW`).

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

Configuring Modbus TCP: Client Mode

This table describes how to configure client mode:

Step	Action
1	In the **Configuration** window, click **ETH1→ Modbus TCP** to display the Modbus TCP properties. The following illustration shows the properties displayed in the editor area: 
2	Add a remote device. Refer to Adding Remote Devices *(see page 147)*.
3	If you want to configure Modbus TCP IOScanner, select **Enable Modbus TCP IOScanner**. **NOTE:** If the **Enable Modbus TCP IOScanner** button is grayed out, verify that the **Functional Level** of your application (**Programming → Tasks → Behavior** tab) is at least **Level 6.0** and that there is no instance configured in **Serial line → Modbus Serial IOScanner**. You can configure and add remote devices for Modbus TCP even if Modbus TCP IOScanner is enabled.

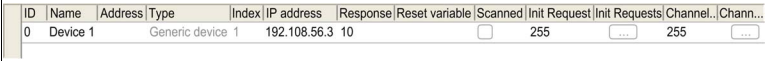
Adding Remote Devices

The following table describes the parameters of **Client mode: remote device table (max 16)** to add a device:

Parameter	Editable(1)	Value	Default Value	Description
IP address	Yes	*w.x.y.z*(2)	–	Allows you to specify the IP address of the device to add. Also, refer to Adding Remote Devices.
Generic **Drive** **Predefined**	Yes	Selection	Generic	Allows you to select the type of device to add. **Drive** and **Predefined** are available if Modbus TCP IOScanner is enabled.

(1) Only if the **Modbus server enabled** option is selected in the **Security Parameters** section of the Ethernet properties window *(see page 137)*.
(2) *w*, *x*, *y*, and *z* are the bytes that store the address and each byte can store a value in the range.

This table describes how to add a remote device:

Step	Action
1	Enter the IP address in the **IP address** field.
2	Select **Generic**, **Drive**, or **Predefined**. **Drive** and **Predefined** are only enabled if **Enable Modbus TCP IOScanner** is selected.
3	Click the **Add** button. The **Add** button is disabled if: ● The maximum of 16 devices is already configured. ● The IP address is in an incorrect format. **Result**: A list of remote devices that you have added appears on the screen. 
4	Click **Apply**.

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

Parameter	Editable	Value	Default Value	Description
ID	No	0...15	**0**	Unique device identifier assigned by SoMachine Basic.

(1) *w*, *x*, *y*, and *z* are the bytes that store the address and each byte can store a value in the range 0...255.
(2) *x* and *n* are integers respectively incremented each time a device or a drive device is added.
(3) Enabled if **Modbus Serial IOScanner** is not configured in **Serial line** node →**Protocol Settings**.

Parameter	Editable	Value	Default Value	Description
Name	Yes	1...32 characters The device name must be unique.	**Device x** (1)	The name of the device.
Address	No	– %DRVn(2)	– %DRVn	%DRVn is used to configure the device in the application using Drive function blocks.
Type	No	Type of the device	–	To change the device type, you must remove the device from the list (by right-clicking and choosing **Delete**), then add the correct device type.
Index	No	1...16	–	The index number of the devices which are remotely connected.
IP address	Yes	*w.x.y.z*(2)	–	Address used to identify the device within the network. Duplicate slave addresses are allowed.
Response timeout (x 100 ms)	Yes	0...65535	10	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. At the end of this period, if a TCP connection is still not established, the controller stops connection attempts until the next connection request with an EXCH instruction.
Reset variable	Yes	%Mn	–	Specify the address of the memory bit to use to reset the device (re-send the initialization requests). When the specified memory bit is set to 1 by the application, the device is reset.
Scanned	No	TRUE/FALSE	TRUE	Allows to see which device is configured for Modbus TCP IOScanner.

(1) *w*, *x*, *y*, and *z* are the bytes that store the address and each byte can store a value in the range 0...255.
(2) *x* and *n* are integers respectively incremented each time a device or a drive device is added.
(3) Enabled if **Modbus Serial IOScanner** is not configured in **Serial line** node →**Protocol Settings**.

Parameter	Editable	Value	Default Value	Description
Init Request Unit ID	Yes	0...255	255	Specify the unit ID of the local device. Modbus TCP requests originating from a device with the same unit ID are sent to the Modbus mapping table instead of the regular Modbus server.
Init. requests (3)	Yes	...	–	Click to display the Initialization request assistant window *(see page 149)*.
Channels Unit ID	Yes	0...255	255	Specify the unit ID of the local device. Modbus TCP requests originating from a device with the same unit ID are sent to the Modbus mapping table instead of the regular Modbus server.
Channels (3)	Yes	...	–	Click to display the Channel assistant window *(see page 151)*.

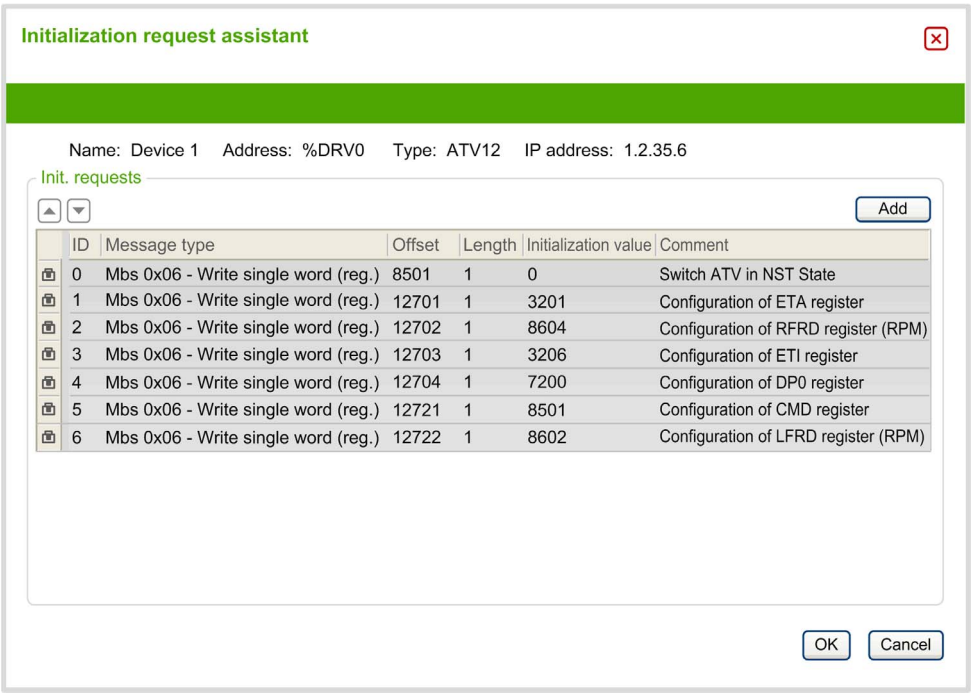
(1) *w*, *x*, *y*, and *z* are the bytes that store the address and each byte can store a value in the range 0...255.
(2) *x* and *n* are integers respectively incremented each time a device or a drive device is added.
(3) Enabled if **Modbus Serial IOScanner** is not configured in **Serial line** node →**Protocol Settings**.

Configuring Initialization Requests

Initialization requests are device-specific commands sent by the Modbus TCP IOScanner or Modbus Serial IOScanner to initialize a slave device. The Modbus TCP IOScanner or Modbus Serial IOScanner does not start cyclic data exchange with the device until all its initialization requests have been acknowledged by the device. During the initialization phase, network objects are not updated.

Up to 20 initialization requests can be defined for each slave device.

The **Initialization request assistant** window presents the defined initialization requests:



Preconfigured initialization requests are displayed with a lock symbol and a gray background. Some parameters cannot be modified for predefined initialization requests.

According to the device type that you selected, some initialization requests may be configured.

This table describes the properties of initialization requests:

Parameter	Editable	Value	Default Value	Description
ID	No	0...19	**0**	Unique initialization request identifier.
Message type	Yes, if initialization request is not predefined.	See Supported Modbus Function Codes *(see page 195)*	**Mbs 0x05 - Write single bit (coil)**	Select the Modbus function code for the type of exchange to use for this initialization request. **NOTE:** If configuring a generic device that does not support the default **Mbs 0x05 - Write single bit (coil)** request type, you must replace the default value with a supported request type.

Parameter	Editable	Value	Default Value	Description
Offset	Yes, if initialization request is not predefined.	0...65535	0	Offset of the first register to initialize.
Length	Yes, if initialization request is not predefined.	1 for **Mbs 0x05 - Write single bit (coil)** 1 for **Mbs 0x06 - Write single word (register)** 128 for **Mbs 0x0F - Write multiple bits (coils)** 123 for **Mbs 0x10 - Write multiple words (reg.)**	1	Number of objects (memory words or bits) to be initialized. For example, if writing multiple words with **Offset** = 2 and **Length** = 3, `%MW2`, `%MW3`, and `%MW4` are initialized.
Initialization value	Yes, if initialization request is not predefined.	0...65535 if memory words (registers) are being initialized 0...1 if memory bits (coils) are being initialized	0	Value to initialize the targeted registers with.
Comment	Yes, if initialization request is not predefined.	-	Empty	Optionally, type a comment to associate with this request.

Click **Add** to create new initialization requests.

Select an entry then use the up arrow and down arrow buttons to change the order in which the initialization requests are sent to the device.

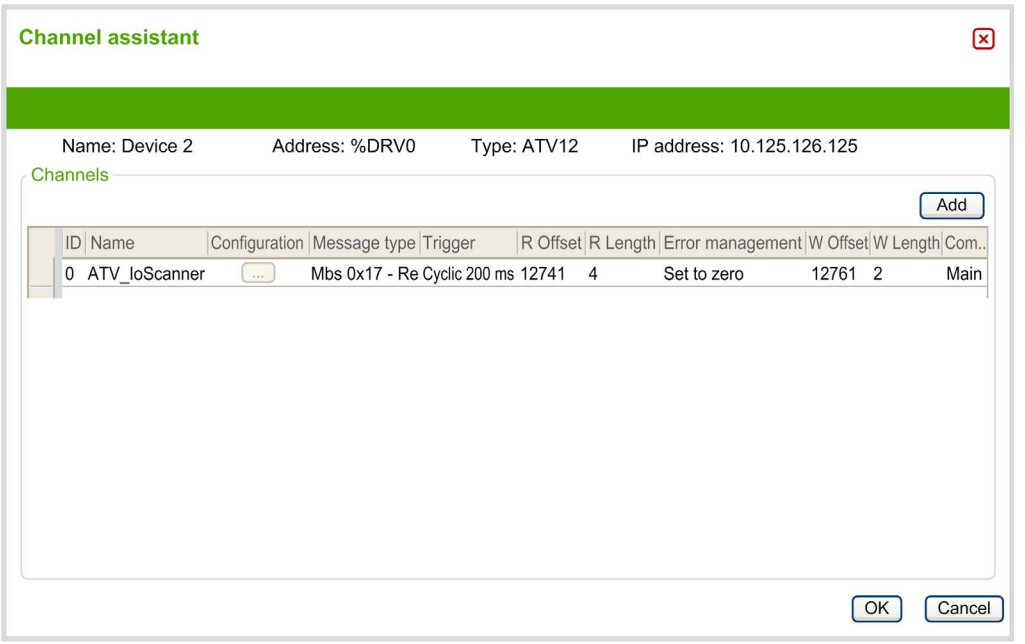
When the initialization requests have been defined, click **OK** to save the configuration and close the **Initialization request assistant**.

Channel Assistant

Up to 10 channels can be defined for each slave device. Each channel represents a single Modbus request.

NOTE: The number of objects defined (items of data read and written) is validated when you click **Apply** on the properties window.

The **Channel assistant** window lists the defined channels:



Preconfigured channels are displayed with a lock symbol and a gray background. Some parameters cannot be modified for predefined channels.

This table describes the properties of channels:

Parameter	Editable	Value	Default Value	Description
ID	No	0...19	**0**	Unique initialization identifier.
Name	Yes	0...32 characters	Device_channel0	Double-click to edit the name of the channel.
Configuration	Yes	...	-	Click to display the Channel Assistant window.
Message type	No	-	-	The Modbus function code that was selected in the Channel Assistant window.
Trigger	No	-	-	The trigger type and cycle time that was selected in the Channel Assistant window.
R Offset	No	-	-	The READ object offset that was selected in the Channel Assistant window.

Parameter	Editable	Value	Default Value	Description
R Length	No	-	-	The READ object length that was selected in the Channel Assistant window.
Error management	No	-	-	The error management policy that was selected in the Channel Assistant window.
W Offset	No	-	-	The WRITE object offset that was selected in the Channel Assistant window.
W Length	No	-	-	The WRITE object length that was selected in the Channel Assistant window.
Comment	Yes	-	Empty	Optionally, type a comment to associate with this channel.

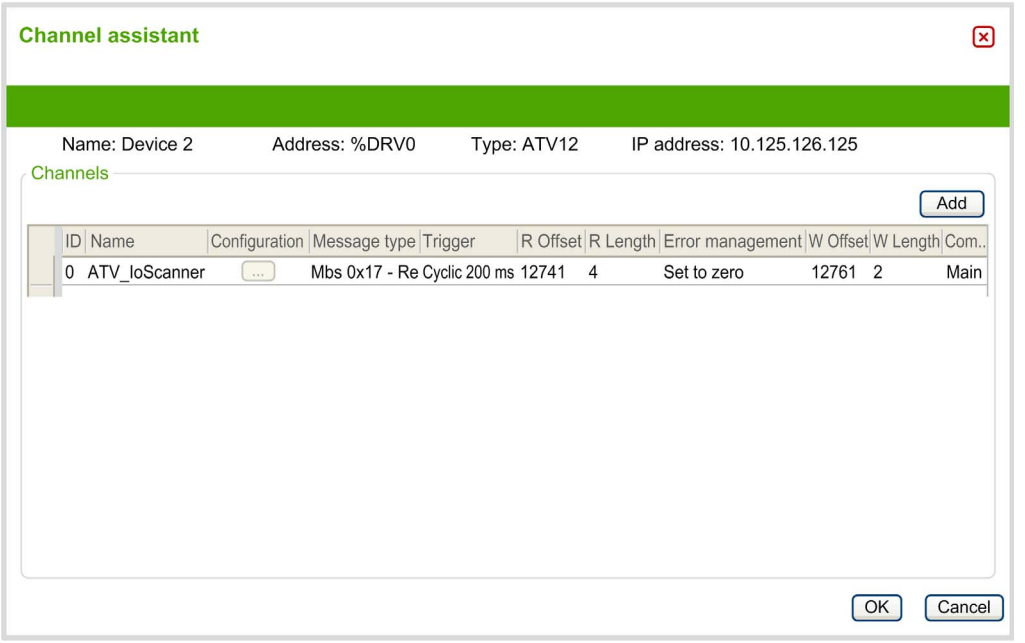
Click **Add** to create a new channel.

When the channels have been defined, click **OK** to save the configuration and close the **Channel assistant**.

Configuring Channels

Use the **Channel assistant** window to configure channels.

The following example shows a channel configured for a Read/Write Multiple Words request (Modbus function code 23). It reads one word from the register with offset 16#0C21 and writes two words to the register with offset 16#0C20. This request is executed when there is a rising edge of the defined **Trigger** (see table below):



This table describes the properties of channels:

Parameter	Editable	Value	Default Value	Description
Name	Yes	0...32 characters	**Device 0_Channel0**	Enter a name for the channel.
Message type	Yes	See Supported Modbus Function Codes *(see page 195)*	**Mbs 0x17 - Read/Write mult. words (reg.)**	Select the Modbus function code for the type of exchange to use on this channel.

Parameter	Editable	Value	Default Value	Description
Trigger	Yes	**Cyclic** **Rising edge**	**Cyclic**	Choose the trigger type for the data exchange: ● **Cyclic**: The request is triggered with the frequency defined in the **Cycle Time (x 10 ms)** field ● **Rising edge**: The request is triggered upon detection of a rising edge of a memory bit. Specify the address of the **Memory bit** to use.
Cycle time (x 10 ms) (If **Cyclic** is selected)	Yes	1...6000	20	Specify the periodic trigger cycle time, in units of 10 ms.
Memory bit (If **Rising edge** is selected)	Yes	%Mn	-	Specify a memory bit address, for example, %M8. The data exchange is triggered when a rising edge of this memory bit is detected.
Comment	Yes	-	Empty	Optionally, type a comment to describe the purpose of the channel.
READ objects				
Offset	Yes	0...65535	0	Address of the first memory word (register) or bit (coil) to read.
Length	Yes	See Supported Modbus Function Codes *(see page 195)* for maximum length	-	Number of memory words (registers) or bits (coils) to read.
Error management	Yes	**Set to zero** **Retain last value**	**Set to zero**	Specify how to manage the situation when data can no longer be read from the device: ● Select **Set to zero** to set the last data values received to zero. ● Select **Retain last value** to keep the last data values received.
WRITE objects				
Offset	Yes	0...65535	0	Address of the first memory word (register) or bit (coil) to write.

Parameter	Editable	Value	Default Value	Description
Length	Yes	See Supported Modbus Function Codes *(see page 195)* for maximum length	-	Number of memory words (registers) or bits (coils) to write.

Click **OK** to complete channel configuration.

Configuring EtherNet/IP

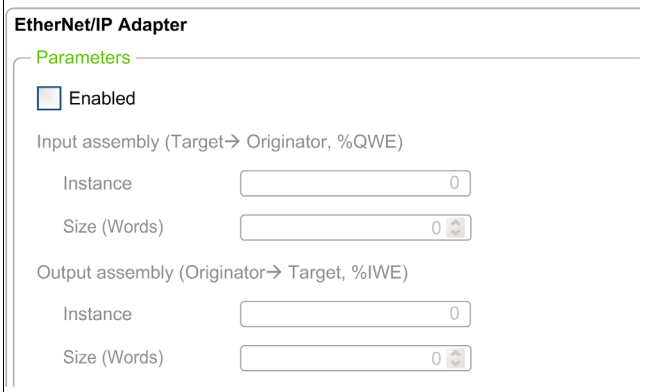
Introduction

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

For further information about EtherNet/IP, refer to *www.odva.org*

EtherNet/IP Adapter Configuration

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

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

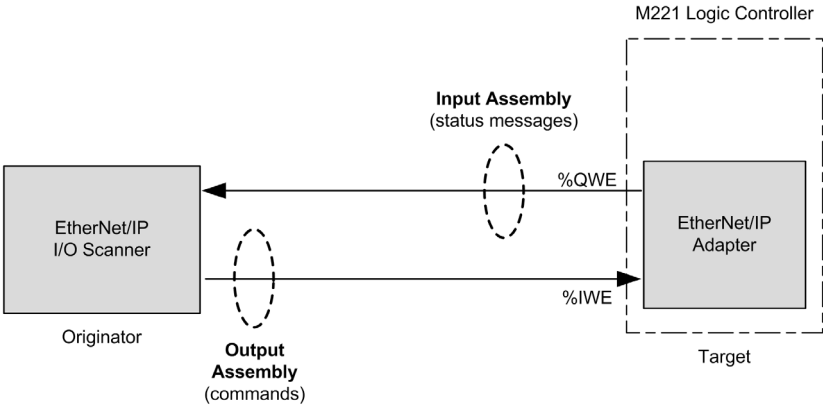
EtherNet/IP Adapter Properties

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

Parameter	Editable	Value	Default Value	Description
Enabled	Yes	TRUE/FALSE	FALSE	Select to enable the EtherNet/IP Adapter configuration. **NOTE:** If you deselect the **Enabled** 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 **Security Parameters** for the protocol in the Ethernet properties window *(see page 137)*. When disabled, by deselecting the **Enabled** checkbox, configured fallback values *(see page 226)* of `%QWE` objects, as well as symbols and comments, are lost.
Input assembly (Target -->Originator, %QWE)				
Instance	Yes	1...255	100	The identifier of the Input assembly.
Size (words)	Yes	1...20	20	The size of the Input assembly.
Output assembly (Originator-->Target, %IWE)				
Instance	Yes	1...255	150	The identifier of the Output assembly.
Size (words)	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.

Profile

The controller supports the following objects:

Object class	Class ID (hex)	Cat.	Number of Instances	Effect on Interface Behavior
Identity Object *(see page 160)*	01	1	1	Provides the identification of the device and general information about it. Supports the reset service.
Message Router Object *(see page 163)*	02	1	1	Provides a message connection that allows the client to address a service to any object class or instance residing in the device.
Assembly Object *(see page 166)*	04	2	2	Binds attributes of multiple objects, which allows data to or from each object to be sent or received over a single connection.
Connection Manager Object *(see page 168)*	06	–	1	Manages the characteristics of a communication connection.

Object class	Class ID (hex)	Cat.	Number of Instances	Effect on Interface Behavior
TCP/IP Interface Object *(see page 171)*	F5	1	1	Provides the mechanism to configure the TCP/IP network interface of a device.
Ethernet Link Object *(see page 173)*	F6	1	1	Maintains link specific counters and status information for an IEEE 802.3 communication interface.

Identity Object (Class ID = 01 hex)

The following table describes the class attributes of the Identity Object (Instance 0):

Attribute ID	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Implementation revision of the Identity Object
2	Get	Max Instances	UINT	01	The largest instance number
3	Get	Number of Instances	UINT	01	The number of object instances
4	Get	Optional Instance Attribute List	UINT, UINT [ ]	00	The first 2 bytes contain the number of optional instance attributes. Each following pair of bytes represents the number of other optional instance attributes.
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	07	The largest instance attributes value

The following table describes the Class Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
05	Reset (1)	Initializes EtherNet/IP component (controller reboot)
0E	Get Attribute Single	Returns the value of the specified attribute

(1) Reset Service description:
When the Identity Object receives a Reset request, it:
- determines whether it can provide the type of reset requested
- responds to the request
- attempts to perform the type of reset requested

The Reset common service has one specific parameter, Type of Reset (USINT), with the following values:

Value	Type of Reset
0	Reboot the controller **NOTE:** This value is the default value if this parameter is omitted.
1	Reset Warm
2	Not supported
3...99	Reserved
100...199	Not used
200...255	Reserved

The following table describes the Instance attributes:

Attribute ID	Access	Name	Data Type	Value (hex)	Details
1	Get	Vendor ID	UINT	F3	Schneider Automation identifier
2	Get	Device type	UINT	0E	Device is a logic controller
3	Get	Product code	UINT	1003	M221 Logic Controller product code
4	Get	Revision	Struct of USINT, USINT	–	Product revision of the controller.(1) Equivalent to the 2 low bytes of the controller version. **Example:** For the M221 Logic Controller firmware version 1.3.2.0, the value read is **1.3**
5	Get	Status	WORD (1)	–	See definition in the table below

(1) Mapped in a WORD:
- MSB: minor revision (second USINT)
- LSB: major revision (first USINT)

Attribute ID	Access	Name	Data Type	Value (hex)	Details
6	Get	Serial number	UDINT	–	Serial number of the controller XX + 3 least significant bytes of the MAC address
7	Get	Product name	Struct of USINT, STRING	–	Maximum length is 32. Example: TM221CE16T
(1) Mapped in a WORD: ● MSB: minor revision (second USINT) ● LSB: major revision (first USINT)					

Status Description (Attribute 5):

Bit	Name	Description
0	Owned	Unused
1	Reserved	–
2	Configured	TRUE indicates the device application has been reconfigured.
3	Reserved	–
4...7	Extended Device Status	● 0: self-testing or undetermined ● 1: firmware update in progress ● 2: at least one invalid I/O connection error detected ● 3: no I/O connections established ● 4: non-volatile configuration invalid ● 5: non recoverable error detected ● 6: at least one I/O connection in RUNNING state ● 7: at least one I/O connection established, all in idle mode ● 8: reserved ● 9...15: unused
8	Minor Recoverable Error	TRUE indicates the device detected an error, which, under most circumstances, is recoverable. This type of event does not lead to a change in the device state.
9	Minor Unrecoverable Error	TRUE indicates the device detected an error, which, under most circumstances, is not recoverable. This type of event does not lead to a change in the device state.
10	Major Recoverable Error	TRUE indicates the device detected an error, which requires the device to report an exception and enter into the HALT state. This type of event leads to a change in the device state, but, under most circumstances, is recoverable.
11	Major Unrecoverable Error	TRUE indicates the device detected an error, which requires the device to report an exception and enter into the HALT state. This type of event leads to a change in the device state, but, under most circumstances is not recoverable.
12...15	Reserved	–

Message Router Object (Class ID = 02 hex)

The following table describes the class attributes of the Message Router Object (Instance 0):

Attribute ID	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Implementation revision of the Message Router Object
2	Get	Max Instances	UINT	01	The largest instance number
3	Get	Number of Instance	UINT	01	The number of object instances
4	Get	Optional Instance Attribute List	Struct of UINT, UINT [ ]	–	The first 2 bytes contain the number of optional instance attributes. Each following pair of bytes represents the number of other optional instance attributes (from 100 to 119).
5	Get	Optional Service List	UINT	00	The number and list of any implemented optional services attribute (0: no optional services implemented)
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	77	The largest instance attributes value

NOTE: Use instance 0 to read the Class Attributes information.

The following table describes the Class Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services (Instance 1):

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance attributes:

Attribute ID	Access	Name	Data Type	Value (hex)	Description
1	Get	Implemented Object List	Struct of UINT, UINT [ ]	–	Implemented Object list. The first 2 bytes contain the number of implemented objects. Each following pair of bytes represents another implemented class number. This list contains the following objects: ● 01: Identity ● 02: Message Router ● 04: Assembly ● 06: Connection Manager ● F5: TCP/IP ● F6: Ethernet Link
2	Get	Number available	UINT	08	Maximum number of concurrent CIP (Class 1 or Class 3) connections supported
100	Get	Total incoming Class1 packets received during the last second	UINT	–	Total number of incoming packets received for all implicit (Class 1) connections during the last second
101	Get	Total outgoing Class1 packets sent during the last second	UINT	–	Total number of outgoing packets sent for all implicit (Class 1) connections during the last second
102	Get	Total incoming Class3 packets received during the last second	UINT	–	Total number of incoming packets received for all explicit (Class 3) connections during the last second
103	Get	Total outgoing Class3 packets sent during the last second	UDINT	–	Total number of outgoing packets sent for all explicit (Class 3) connections during the last second
104	Get	Total incoming unconnected packets received during the last second	UINT	–	Total number of incoming unconnected packets received during the last second
105	Get	Total outgoing unconnected packets sent during the last second	UINT	–	Total number of outgoing unconnected packets sent during the last second

Attribute ID	Access	Name	Data Type	Value (hex)	Description
106	Get	Total incoming EtherNet/IP packets received during the last second	UINT	–	Total unconnected Class 1 or Class 3 packets received during the last second
107	Get	Total outgoing EtherNet/IP packets sent during the last second	UINT	–	Total unconnected Class 1 or Class 3 packets sent during the last second
108	Get	Total incoming Class1 packets received	UINT	–	Total number of incoming packets received for all implicit (Class 1) connections
109	Get	Total outgoing Class1 packets sent	UINT	–	Total number of outgoing packets sent for all implicit (Class 1) connections
110	Get	Total incoming Class3 packets received	UINT	–	Total number of incoming packets received for all explicit (Class 3) connections. This number includes the packets that would be returned if an error had been detected (listed in the next two rows).
111	Get	Total incoming Class3 packets Invalid Parameter Value	UINT	–	Total number of incoming Class 3 packets that targeted an unsupported service/class/instance/attribute/member
112	Get	Total incoming Class3 packets Invalid Format	UINT	–	Total number of incoming Class 3 packets that had an invalid format
113	Get	Total outgoing Class3 packets sent	UINT	–	Total number of packets sent for all explicit (Class 3) connections
114	Get	Total incoming unconnected packets received	UINT	–	Total number of incoming unconnected packets. This number includes the packets that would be returned if an error had been detected (listed in the next two rows).

Attribute ID	Access	Name	Data Type	Value (hex)	Description
115	Get	Total incoming unconnected packets Invalid Parameter Value	UINT	–	Total number of incoming unconnected packets that targeted an unsupported service/class/instance/attribute/member
116	Get	Total incoming unconnected packets Invalid Format	UINT	–	Total number of incoming unconnected packets that had an invalid format
117	Get	Total outgoing unconnected packets sent	UINT	–	Total number of all unconnected packets sent
118	Get	Total incoming EtherNet/IP packets	UINT	–	Total number of unconnected (Class 1) or Class 3 packets received
119	Get	Total outgoing EtherNet/IP packets	UINT	–	Total number of unconnected (Class 1) or Class 3 packets sent

Assembly Object (Class ID = 04 hex)

The following table describes the class attributes of the Assembly Object (Instance 0):

Attribute ID	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	02	Implementation revision of the Assembly Object
2	Get	Max Instances	UINT	–	The largest instance number of created objects of this class. **Example:** If input instances = 200, output instances = 100, this attribute returns 200.
3	Get	Number of Instances	UINT	02	The number of object instances
4	Get	Optional Instance Attribute List	Struct of: UINT UINT [ ]	–	The first 2 bytes contain the number of optional instance attributes. Each following pair of bytes represents the number of other optional instance attributes.
5	Get	Optional Service List	UINT	00	The number and list of any implemented optional services attribute (0: no optional services implemented)

Attribute ID	Access	Name	Data Type	Value (hex)	Details
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	04	The largest instance attributes value

The following table describes the Class Services:

Service Code (hex)	Name	Description
0E	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

Service Code (hex)	Name	Description
0E	Get Attribute Single	Returns the value of the specified attribute
10	Set Attribute Single	Modifies the value of the specified attribute
18	Get Member	Reads a member of an Assembly object instance
19	Set Member	Modifies a member of an Assembly object instance

Instances Supported

Output means OUTPUT from Originator controller (= %IWE for the M221 Logic Controller).

Input means INPUT from Originator controller (= %QWE for the M221 Logic Controller).

The controller supports 2 Assemblies:

Name	Instance	Data Size
Input Assembly (EtherNet/IP) (%QWE)	Configurable from 1...255	1...20 words
Output Assembly (EtherNet/IP) (%IWE)	Configurable from 1...255	1...20 words

NOTE: The Assembly object binds together the attributes of multiple objects so that information sent to or received from each object can be communicated over a single connection. Assembly objects are static.
The assemblies in use can be modified through the parameter access of the network configuration tool (RSNetWorx). You must perform a power cycle of the logic controller to register a new assembly assignment.

The following table describes the Instance attributes:

Attribute ID	Access	Name	Data Type	Value	Description
1	Get	Number of Member Object List	UINT	1...20	Number of members for this assembly
2	Get	Member List	ARRAY of Struct	–	Array of 1 structure where each structure represents one member
3	Get/Set	Instance Data	ARRAY of Byte	–	Data Set service only available for Controller output
4	Get	Instance Data Size	UINT	2...40	Size of data in bytes

Member list content:

Name	Data Type	Value	Type of Reset
Member data size	UINT	4...40	Member data size in bits
Member path size	UINT	6	Size of the EPATH (see table below)
Member path	EPATH	–	EPATH to the Member

EPATH is:

Word	Value (hex)	Semantic
0	2004	Class 4
1	24xx	Instance xx, where xx is the instance value (for example: 2464 hex = instance 100)
2	xxxx	See the Common Industrial Protocol Specification Volume 1 - Appendix C for the format of this field

Connection Manager Object (Class ID = 06 hex)

The following table describes the class attributes of the Assembly Object (Instance 0):

Attribute ID	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	01	Implementation revision of the Connection Manager Object
2	Get	Max Instances	UINT	01	The largest instance number
3	Get	Number of Instances	UINT	01	The number of object instances

Attribute ID	Access	Name	Data Type	Value (hex)	Details
4	Get	Optional Instance Attribute List	Struct of: UINT UINT [ ]	–	The number and list of the optional attributes. The first word contains the number of attributes to follow and each following word contains another attribute code. Following optional attributes include: ● Total number of incoming connection open requests ● The number of requests rejected because of the non-conforming format of the Forward Open ● The number of requests rejected because of insufficient resources ● The number of requests rejected because of the parameter value sent with the Forward Open ● The number of Forward Close requests received ● The number of Forward Close requests that had an invalid format ● The number of Forward Close requests that could not be matched to an active connection ● The number of connections that have timed out because the other side stopped producing, or a network disconnection occurred
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	08	The largest instance attributes value

The following table describes the Class Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

The following table describes the Instance Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all instance attributes
0E	Get Attribute Single	Returns the value of the specified attribute
4E	Forward Close	Closes an existing connection
52	Unconnected Send	Sends a multi-hop unconnected request
54	Forward Open	Opens a new connection

The following table describes the Instance attributes (Instance 1):

Attribute ID	Access	Name	Data Type	Value	Description
1	Get	Open Requests	UINT	–	Number of Forward Open service requests received
2	Get	Open Format Rejects	UINT	–	Number of Forward Open service requests which were rejected due to invalid format
3	Get	Open Resource Rejects	UINT	–	Number of Forward Open service requests which were rejected due to lack of resources
4	Get	Open Other Rejects	UINT	–	Number of Forward Open service requests which were rejected for reasons other than invalid format or lack of resources
5	Get	Close Requests	UINT	–	Number of Forward Close service requests received
6	Get	Close Format Requests	UINT	–	Number of Forward Close service requests which were rejected due to invalid format
7	Get	Close Other Requests	UINT	–	Number of Forward Close service requests which were rejected for reasons other than invalid format
8	Get	Connection Timeouts	UINT	–	Total number of connection timeouts that have occurred in connections controlled by this Connection Manager

TCP/IP Interface Object (Class ID = F5 hex)

This object provides the mechanism to configure a TCP/IP network interface device.

The following table describes the class attributes of the TCP/IP Interface Object (Instance 0):

Attribute ID	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	02	Implementation revision of the TCP/IP Interface Object
2	Get	Max Instances	UINT	01	The largest instance number
3	Get	Number of Instance	UINT	01	The number of object instances
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	06	The largest instance attributes value

The following table describes the Class Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

Instance Codes

Only instance 1 is supported.

The following table describes the Instance Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all instance attributes
0E	Get Attribute Single	Returns the value of the specified instance attribute

The following table describes the Instance Attributes (Instance 1):

Attribute ID	Access	Name	Data Type	Value	Description
1	Get	Status	DWORD	Bit level	• 0: The interface configuration attribute has not been configured. • 1: The interface configuration contains a valid configuration. All other bits are reserved and set to 0.
2	Get	Configuration Capability	DWORD	Bit level	• 0: BOOTP Client • 2: DHCP Client All other bits are reserved and set to 0.

Attribute ID	Access	Name	Data Type	Value	Description
3	Get	Configuration	DWORD	Bit level	• 0: The interface configuration is valid. • 1: The interface configuration is obtained with BOOTP. • 2: The interface configuration is obtained with DHCP. • 3: Reserved • 4: DNS Enable All other bits are reserved and set to 0.
4	Get	Physical Link	UINT	Path size	Number of 16-bit words in the Path element
			Padded EPATH	Path	Logical segments identifying the physical link object. The path is restricted to one logical class segment and one logical instance segment. The maximum size is 12 bytes.
5	Get	Interface configuration	UDINT	IP Address	Hexadecimal format Example: 55 DD DD DE = 85.221.221.222
			UDINT	Network Mask	Hexadecimal format Example: FF 0 0 0 = 255.0.0.0
			UDINT	Gateway Address	Hexadecimal format Example: 55 DD DD DE = 85.221.221.222
			UDINT	Primary Name	0: no primary name server address has been configured.
			UDINT	Secondary Name	0: no secondary name server address has been configured. Otherwise, the name server address shall be set to a valid Class A, B, or C address.
			STRING	Default Domain Name	ASCII characters. Maximum length is 16 characters. Padded to an even number of characters (padding not included in length). 0: no Domain Name is configured
6	Get	Host Name	UINT	–	Host name length
			STRING	–	ASCII characters. Maximum length is 64 characters. Padded to an even number of characters (padding not included in length). 0: no Host Name is configured

Ethernet Link Object (Class ID = F6 hex)

This object maintains link specific counters and status information for an Ethernet 802.3 communications interface.

The following table describes the class attributes of the Ethernet Link Object (Instance 0):

Attribute ID	Access	Name	Data Type	Value (hex)	Details
1	Get	Revision	UINT	03	Implementation revision of the Ethernet Link Object
2	Get	Max Instances	UINT	01	The largest instance number
3	Get	Number of Instances	UINT	01	The number of object instances
6	Get	Max Class Attribute	UINT	07	The largest class attributes value
7	Get	Max Instance Attribute	UINT	03	The largest instance attribute value

The following table describes the Class Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all class attributes
0E	Get Attribute Single	Returns the value of the specified attribute

Instance Codes

Only instance 1 is supported.

The following table describes the Instance Services:

Service Code (hex)	Name	Description
01	Get Attribute All	Returns the value of all instance attributes
0E	Get Attribute Single	Returns the value of the specified instance attribute

The following table describes the Instance Attributes (Instance 1):

Attribute ID	Access	Name	Data Type	Value	Description
1	Get	Interface Speed	UDINT	–	Speed in Mbps (10 or 100)
2	Get	Interface Flags	DWORD	Bit level	• 0: link status • 1: half/full duplex • 2...4: negotiation status • 5: manual setting / requires reset • 6: local hardware error detected All other bits are reserved and set to 0.
3	Get	Physical Address	ARRAY of 6 USINT	–	This array contains the MAC address of the product. Format: XX-XX-XX-XX-XX-XX

Section 6.2
Serial Line Configuration

What Is in This Section?

This section contains the following topics:

Configuring Serial Lines

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)

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

- Modbus (RTU or ASCII) *(see page 180)*. Serial lines are configured for the Modbus RTU protocol by default.
- ASCII *(see page 180)*
- Modbus Serial IOScanner *(see page 185)*. Only one instance can be configured: if configured on one serial line, it cannot be used on the other serial line.
  NOTE: Care must be taken when both the Modbus Serial IOScanner and Message (%MSG) function blocks *(see SoMachine Basic, Generic Functions Library Guide)* are used in your application, as this can lead to the cancellation of on-going IOScannner communication.

  The application must be configured with a functional level *(see SoMachine Basic, Operating Guide)* of at least **Level 5.0** to support the Modbus Serial IOScanner.

NOTE: The TMH2GDB Remote Graphic Display *(see page 184)* protocol can only be configured on SL1.

Modem Support

A modem connection allows you to:

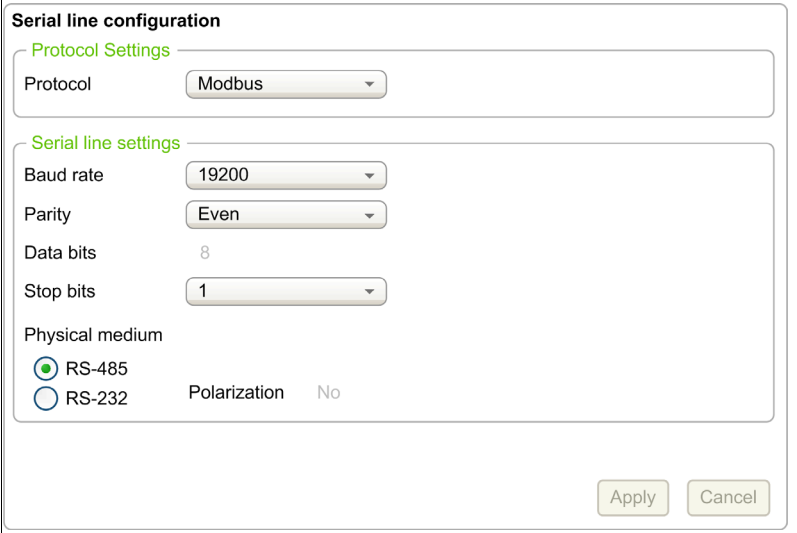
- Remotely access the controller for the purpose of programming and/or monitoring. In this case, a local modem must be connected to the PC running the SoMachine Basic software, and a modem connection must be configured *(see SoMachine Basic, Operating Guide)*.
- Perform data exchanges between controllers using the Modbus protocol.
- Send or receive messages with any device using the `Send Receive Message` function block.
- Send or receive SMS to or from mobile phone or other devices able to send or receive SMS messages.

Serial lines support the following features to simplify modem connections:

- An initialization (Init) command to send an initial configuration to the modem. This command is automatically sent by the controller after an application download or at power on.
- System bit `%S105` to be able to send the Init command to the modem again.
- System word `%SW167` to provide the status of the Init command operation.

Serial Line Configuration

This table describes how to configure the serial line:

Step	Action
1	Click the **SL1 (Serial line)** or **SL2 (Serial line)** node in the hardware tree to display the serial line configuration. 
2	Select the **Protocol** to use on the serial line. For detailed information on the serial line configuration parameters, refer to the following table.
3	Click **Apply**.
4	In the hardware tree select the **Modbus**, **ASCII**, **Display**, or **Modbus Serial IOScanner** node that appears below the **SL1 (Serial Line)** or **SL2 (Serial Line)** node.

This table describes protocol and serial line settings of the serial line:

Parameter	Editable	Value	Default Value	Description
Protocol Settings				
Protocol	Yes	**Modbus** **ASCII** **TMH2GDB** **Modbus Serial** **IOScanner**	**Modbus**	Select a protocol from the drop-down list. **NOTE:** When using an **SR2MOD03** modem and the `Send Receive SMS` function block, select the **ASCII** protocol.
Serial Line Settings				
Baud rate	Yes	**1200** **2400** **4800** **9600** **19200** **38400** **57600** **115200**	**19200**	Allows you to select the data transmission rate (bits per second) from the drop-down list.
Parity	Yes	**None** **Even** **Odd**	**Even**	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.
Data bits	Yes (only for the **ASCII** protocol	**7** **8**	**8**	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.
Stop bits	Yes	**1** **2**	**1**	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
Physical medium	Yes	**RS-485** **RS-232**	**RS-485**	Allows you to select the physical medium for communication. You can select either **RS-485** or **RS-232** medium. For the embedded serial line 2, only **RS-485** 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. **NOTE:** When using an **SR2MOD03**, select the **RS-232** option.
Polarization	Yes (for cartridges only) No (for the controller)	**Yes** **No**	**No**	Polarization resistors are integrated in cartridge modules. For the controller, this parameter is disabled and for the cartridges, this parameter allows you to switch on or off polarization.

Configuring Modbus and ASCII Protocols

Device Settings for Modbus and ASCII Protocols

This table describes the parameters when the **Modbus** or **ASCII** protocol is selected:

Parameter	Editable	Value	Default Value	Description
Device Settings				
Device	Yes	**None** **Generic Modem** **SR2MOD01** **SR2MOD03**	**None**	Select a device from the drop-down list. Select **SR2MOD03** to use the `%SEND_RECV_SMS` function block.
Init command	Yes	-	-	The Init command is a set of Hayes commands sent to the modem connected on the serial line. 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: ● At power on ● If the `%S105` system bit is set to 1. `%SW167` provides the status of the initialization command sent to the modem. A default Init command is used by SoMachine Basic for **SR2MOD03** modem. For more information, refer to the *SR2MOD02 and SR2MOD03 Wireless Modem User Guide* *(see page 10)*. **NOTE:** To use the 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` (see Recv_SMS Function Block *(see SoMachine Basic, Generic Functions Library Guide)*).

Configuring the Modem Init Command

The Init command is a set of Hayes commands sent to initialize 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.

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 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
```

Protocol Settings for Modbus

This table describes the parameters when the **Modbus** protocol is selected:

Parameter	Editable	Value	Default Value	Description
Transmission mode	Yes	**RTU** **ASCII**	**RTU**	Allows you to select the protocol transmission mode for communication from the drop-down list. Select **ASCII** to use the `%SEND_RECV_SMS` function block. Protocol advanced parameters are displayed based on the selected protocol.
Addressing	Yes	**Slave** **Master**	**Slave**	Allows you to select the addressing mode. You can only select either of the **Slave** or **Master** addressing. Selecting any of the addressing modes clears the present one. A device configured as a slave can send Modbus requests.
Address [1...247]	Yes	1...247	1	Allows you to specify the address ID of the slave. **NOTE:** This field is displayed only for the addressing of the slave. For master, this field does not appear on the screen.

Parameter	Editable	Value	Default Value	Description
Response timeout (x 100 ms)	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.
Time between frames (ms)	Yes	1...255	10	The period of time between frames (corresponds to inter-frame delay used in other products). **NOTE:** The value is subject to adjustment to conform to Modbus standard 3.5 character time delay.

Protocol Settings for ASCII

This table describes the parameters when the **ASCII** protocol is selected:

Parameter	Editable	Value	Default Value	Description
Response timeout (x 100 ms)	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. **NOTE:** When using an **SR2MOD03** and the function block SMS, enter 0 to disable the timeout.
Stop condition				
Frame length received	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. **NOTE:** You can configure only one parameter for stop condition that is either **Frame length received** or **Frame received timeout (ms)**.
Frame received timeout (ms)	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. **NOTE:** When using an **SR2MOD03** and the function block SMS, select the checkbox and enter 200.
Frame structure				
Start character	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.

Parameter	Editable	Value	Default Value	Description
First end character	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. NOTE: To be able to disable the **First end character**, 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.
Second end character	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. NOTE: This field is disabled when the disabled **First end character** is disabled. The ASCII character corresponding to the second end character value is displayed on right-hand side of the value field.
Send frame characters	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.

Configuring the TMH2GDB Remote Graphic Display

Protocol Settings for Display

This table describes the parameters when the **Display** protocol is selected:

Parameter	Editable	Value	Default Value	Description
Time between frames (ms)	Yes	1...255	10	The period of time between frames (corresponds to inter-frame delay used in other products). **NOTE:** The value is subject to adjustment to conform to Modbus standard 3.5 character time delay.

Configuring Modbus Serial IOScanner

Description

Only one instance of IOScanner can be defined: if you configure it on a Ethernet port, you cannot configure it on an serial port. Refer to Configuring Modbus TCP IOScanner.

The maximum number of TCP and Serial IOScanner objects is:

- 128, if the **Functional Level** < **6.0**.
- 512, if the **Functional Level** ≥ **6.0**.

Protocol Settings

This table describes the parameters when the **Modbus Serial IOScanner** protocol is selected:

Parameter	Editable	Value	Default Value	Description
Transmission mode	Yes	**RTU** **ASCII**	**RTU**	Select the protocol transmission mode for communication from the drop-down list.
Response timeout (x 100 ms)	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.
Time between frames (ms)	Yes	1...255	10	The period of time between frames (corresponds to inter-frame delay used in other products). **NOTE:** The value is subject to adjustment to conform to Modbus standard 3.5 character time delay.

Adding a Device on the Modbus Serial IOScanner

Introduction

This section describes how to add devices to be scanned by the Modbus Serial IOScanner.

You can add up to 16 Modbus slave devices.

SoMachine Basic is supplied with a number of predefined device types. Predefined device types have predefined initialization requests and preconfigured channels to facilitate integration of the devices in the network.

A generic slave device is also provided, for which initialization requests and channels must be configured.

Adding a Device on the Modbus Serial IOScanner

To add a device on the Modbus Serial IOScanner:

Step	Action
1	Choose either: ● **Drive** and select one of the supported device types from the dropdown list. ● **Others** and select the device type from the dropdown list. If you cannot find your device type in either list, select **Generic device** and configure it.
2	Click **Add**.
3	Configure the device as described in Device Settings *(see page 186)*.
4	Click **Apply**.

Device Settings

This table describes the parameters when the **Modbus Serial IOScanner** protocol is selected:

Parameter	Editable	Value	Default Value	Description
ID	No	0...15	**0**	Unique device identifier assigned by SoMachine Basic.
Name	Yes	1...32 characters The device name must be unique.	**Device x**(1)	Specify a unique name for the device.
Address	No	– %DRVn (1) (2)	– %DRV0	%DRVn is used to configure the device in the application using Drive function blocks *(see Modicon M221 Logic Controller, Advanced Functions Library Guide)*.

(1) *x* and *n* are integers incremented each time a device or a drive device is added.
(2) Only if **Drive** is selected as the device type.

Parameter	Editable	Value	Default Value	Description
Type	No	Type of the device	–	The device type is not editable. To change the device type, you must remove the device from the list (by right-clicking and choosing **Delete**), then add the correct device type.
Slave address	Yes	1...247	1	Address used to identify the device within the network. Duplicate slave addresses are allowed.
Response timeout (x 100 ms)	Yes	0...255	10	The timeout (in milliseconds) used in data exchanges with the device. This value can be adapted individually to the device and overrides the **Response timeout** set for the master in the **Protocol Settings**.
Reset variable	Yes	%Mn	–	Specify the address of the memory bit to use to reset the device (re-send the initialization requests). When the specified memory bit is set to 1 by the application, the device is reset.
Init. requests	Yes	...	-	Click to display the Initialization request assistant window *(see page 187)*.
Channels	Yes	...	-	Click to display the Channel assistant window *(see page 189)*.

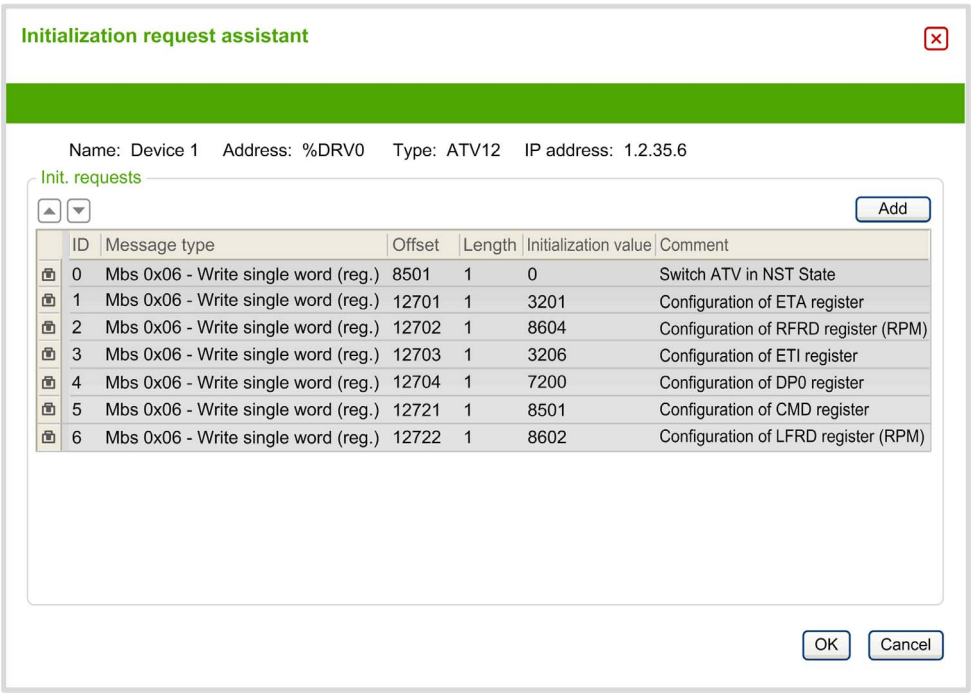
(1) *x* and *n* are integers incremented each time a device or a drive device is added.
(2) Only if **Drive** is selected as the device type.

Configuring Initialization Requests

Initialization requests are device-specific commands sent by the Modbus TCP IOScanner or Modbus Serial IOScanner to initialize a slave device. The Modbus TCP IOScanner or Modbus Serial IOScanner does not start cyclic data exchange with the device until all its initialization requests have been acknowledged by the device. During the initialization phase, network objects are not updated.

Up to 20 initialization requests can be defined for each slave device.

The **Initialization request assistant** window presents the defined initialization requests:



Preconfigured initialization requests are displayed with a lock symbol and a gray background. Some parameters cannot be modified for predefined initialization requests.

According to the device type that you selected, some initialization requests may be configured.

This table describes the properties of initialization requests:

Parameter	Editable	Value	Default Value	Description
ID	No	0...19	**0**	Unique initialization request identifier.
Message type	Yes, if initialization request is not predefined.	See Supported Modbus Function Codes *(see page 195)*	**Mbs 0x05 - Write single bit (coil)**	Select the Modbus function code for the type of exchange to use for this initialization request. **NOTE:** If configuring a generic device that does not support the default **Mbs 0x05 - Write single bit (coil)** request type, you must replace the default value with a supported request type.

Parameter	Editable	Value	Default Value	Description
Offset	Yes, if initialization request is not predefined.	0...65535	0	Offset of the first register to initialize.
Length	Yes, if initialization request is not predefined.	1 for **Mbs 0x05 - Write single bit (coil)** 1 for **Mbs 0x06 - Write single word (register)** 128 for **Mbs 0x0F - Write multiple bits (coils)** 123 for **Mbs 0x10 - Write multiple words (reg.)**	1	Number of objects (memory words or bits) to be initialized. For example, if writing multiple words with **Offset** = 2 and **Length** = 3, `%MW2`, `%MW3`, and `%MW4` are initialized.
Initialization value	Yes, if initialization request is not predefined.	0...65535 if memory words (registers) are being initialized 0...1 if memory bits (coils) are being initialized	0	Value to initialize the targeted registers with.
Comment	Yes, if initialization request is not predefined.	-	Empty	Optionally, type a comment to associate with this request.

Click **Add** to create new initialization requests.

Select an entry then use the up arrow and down arrow buttons to change the order in which the initialization requests are sent to the device.

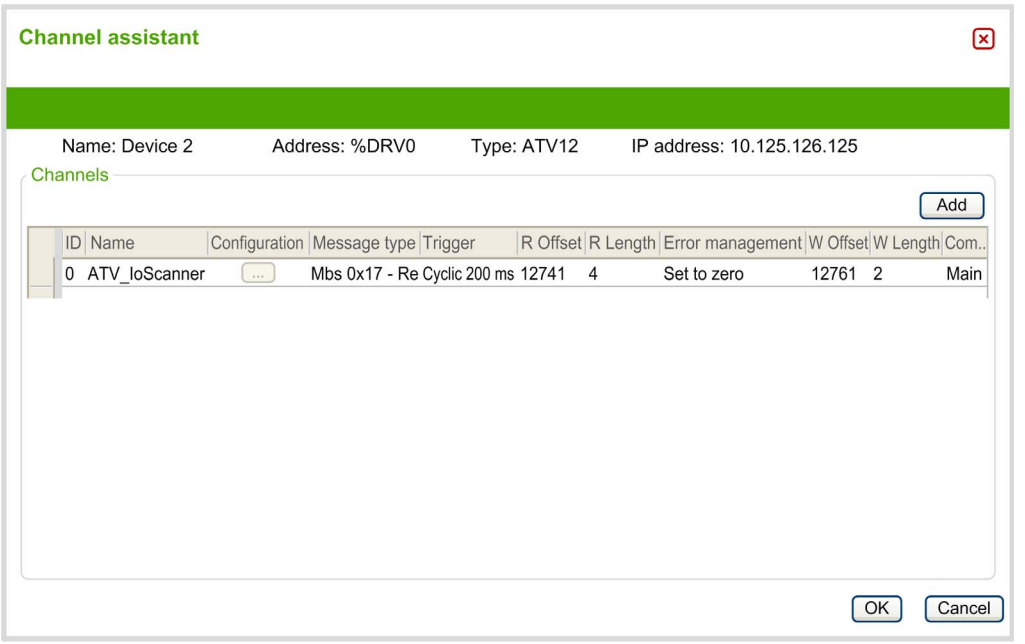
When the initialization requests have been defined, click **OK** to save the configuration and close the **Initialization request assistant**.

Channel Assistant

Up to 10 channels can be defined for each slave device. Each channel represents a single Modbus request.

NOTE: The number of objects defined (items of data read and written) is validated when you click **Apply** on the properties window.

The **Channel assistant** window lists the defined channels:



Preconfigured channels are displayed with a lock symbol and a gray background. Some parameters cannot be modified for predefined channels.

This table describes the properties of channels:

Parameter	Editable	Value	Default Value	Description
ID	No	0...19	**0**	Unique initialization identifier.
Name	Yes	0...32 characters	Device_channel0	Double-click to edit the name of the channel.
Configuration	Yes	...	-	Click to display the Channel Assistant window.
Message type	No	-	-	The Modbus function code that was selected in the Channel Assistant window.
Trigger	No	-	-	The trigger type and cycle time that was selected in the Channel Assistant window.
R Offset	No	-	-	The READ object offset that was selected in the Channel Assistant window.

Parameter	Editable	Value	Default Value	Description
R Length	No	-	-	The READ object length that was selected in the Channel Assistant window.
Error management	No	-	-	The error management policy that was selected in the Channel Assistant window.
W Offset	No	-	-	The WRITE object offset that was selected in the Channel Assistant window.
W Length	No	-	-	The WRITE object length that was selected in the Channel Assistant window.
Comment	Yes	-	Empty	Optionally, type a comment to associate with this channel.

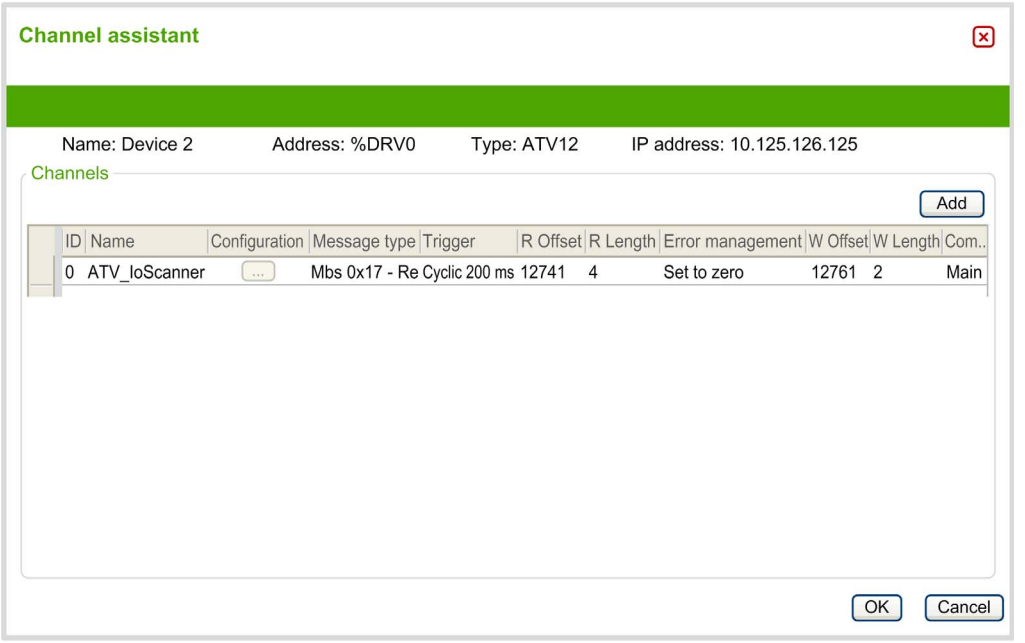
Click **Add** to create a new channel.

When the channels have been defined, click **OK** to save the configuration and close the **Channel assistant**.

Configuring Channels

Use the **Channel assistant** window to configure channels.

The following example shows a channel configured for a Read/Write Multiple Words request (Modbus function code 23). It reads one word from the register with offset 16#0C21 and writes two words to the register with offset 16#0C20. This request is executed when there is a rising edge of the defined **Trigger** (see table below):



This table describes the properties of channels:

Parameter	Editable	Value	Default Value	Description
Name	Yes	0...32 characters	**Device 0_Channel0**	Enter a name for the channel.
Message type	Yes	See Supported Modbus Function Codes *(see page 195)*	**Mbs 0x17 - Read/Write mult. words (reg.)**	Select the Modbus function code for the type of exchange to use on this channel.

Parameter	Editable	Value	Default Value	Description
Trigger	Yes	**Cyclic** **Rising edge**	**Cyclic**	Choose the trigger type for the data exchange: ● **Cyclic**: The request is triggered with the frequency defined in the **Cycle Time (x 10 ms)** field ● **Rising edge**: The request is triggered upon detection of a rising edge of a memory bit. Specify the address of the **Memory bit** to use.
Cycle time (x 10 ms) (If **Cyclic** is selected)	Yes	1...6000	20	Specify the periodic trigger cycle time, in units of 10 ms.
Memory bit (If **Rising edge** is selected)	Yes	%Mn	-	Specify a memory bit address, for example, %M8. The data exchange is triggered when a rising edge of this memory bit is detected.
Comment	Yes	-	Empty	Optionally, type a comment to describe the purpose of the channel.
READ objects				
Offset	Yes	0...65535	0	Address of the first memory word (register) or bit (coil) to read.
Length	Yes	See Supported Modbus Function Codes *(see page 195)* for maximum length	-	Number of memory words (registers) or bits (coils) to read.
Error management	Yes	**Set to zero** **Retain last value**	**Set to zero**	Specify how to manage the situation when data can no longer be read from the device: ● Select **Set to zero** to set the last data values received to zero. ● Select **Retain last value** to keep the last data values received.
WRITE objects				
Offset	Yes	0...65535	0	Address of the first memory word (register) or bit (coil) to write.
Length	Yes	See Supported Modbus Function Codes *(see page 195)* for maximum length	-	Number of memory words (registers) or bits (coils) to write.

Click **OK** to complete channel configuration.

Section 6.3
Supported Modbus Function Codes

Supported Modbus Function Codes

Presentation

This section lists the supported Modbus function codes and their effect on controller memory variables for:

- Modbus Serial *(see page 194)*
- Modbus Serial IOScanner *(see page 195)*
- Modbus TCP *(see page 195)*
- Modbus TCP IOScanner *(see page 195)*

Modbus Serial

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 Serial IOScanner and Modbus TCP IOScanner

This table lists the Modbus function codes supported by the Modbus Serial IOScanner and Modbus TCP IOScanner:

Function Code Dec (Hex)	Description	Available For Configuration	Maximum Length (Bits)
1 (1 hex)	Read multiple bits (coils)	**Channel**	128
2 (2 hex)	Read multiple bits (discrete inputs)	**Channel**	128
3 (3 hex)	Read multiple words (holding registers)	**Channel**	125
4 (4 hex)	Read multiple words (input registers)	**Channel**	125
5 (5 hex)	Write single bit (coil)	**Channel** **Initialization Value** (default message type for initialization values)	1
6 (6 hex)	Write single word (register)	**Channel** **Initialization Value**	1
15 (0F hex)	Write multiple bits (coils)	**Channel** **Initialization Value**	128
16 (10 hex)	Write multiple words (registers)	**Channel** **Initialization Value**	123
23 (17 hex)	Read/write multiple words (registers)	**Channel** (default message type for channel configuration)	125 (read) 121 (write)

Modbus Mapping Table for Modbus TCP

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 145)*.

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.

You can use the SD card when you want to store data. Refer to Data Logging.

What Is in This Chapter?

This chapter contains the following topics:

File Management Operations

Introduction

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

- Clone management *(see page 202)*: back up the application, firmware, and post configuration (if it exists) of the logic controller
- Firmware management *(see page 204)*: download firmware directly to the logic controller, and load firmware to the Remote Graphic Display
- Application management *(see page 208)*: back up and restore the logic controller application, or copy it to another logic controller of the same reference
- Post configuration management *(see page 210)*: add, change, or delete the post configuration file of the logic controller
- Error log management *(see page 212)*: back up or delete the error log file of the logic controller
- Memory management *(see page 215)*: 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 you remove power to the device, or there is a power outage or communication interruption during the transfer of the application, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer. If there is a power outage or communication interruption during a firmware update, or if an invalid firmware is used, your device will become inoperative. In this case, use a valid firmware and reattempt the firmware update.

NOTICE

INOPERABLE EQUIPMENT

- Do not interrupt the transfer of the application program or a firmware change once the transfer has begun.
- Re-initiate the transfer if the transfer is interrupted for any reason.
- Do not attempt to place the device (logic controller, motion controller, HMI controller or drive) into service until the file 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
/disp/	Remote Graphic Display firmware file	TMH2GDB.mfw
/sys/os	Logic controller firmware file	M221.mfw
/TM3	TM3 analog expansion modules firmware	TM3_Ana.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. Script files must be encoded in ANSI format.

This table describes the supported script commands:

Command	Description
Download	Download a file from the SD card to the controller.
Upload	Upload files contained in controller memory to the SD card.
Delete	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"
```

NOTE: Post configuration files specified in **Upload** or **Delete** commands must have the extension `.cfg` or `.CFG`.
If no post configuration file is specified, or the specified file name does not exist, the default file name `Machine.cfg` is assumed.

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 SD card name must be different from `DATA`, refer to Data Logging.
- 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. **Result:** The clone operation starts automatically and the **SD** LED is illuminated.
3	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** LED flashes and the detected error is logged in `Script.log` file. **NOTE:** 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 `RUNNING` state compared to the `STOPPED` 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. **Result:** The clone operation is in progress. **NOTE:** The **SD** LED is turned on during the operation.
4	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` 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

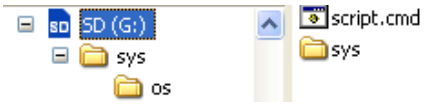
You can use an SD card to download firmware updates directly to the logic controller, a Remote Graphic Display, or TM3 analog expansion modules.

Refer to Controller States and Behavior *(see page 54)* for information on the logic controller operating states and status of the LEDs.

To perform firmware management, the SD card name must be different from `DATA`, refer to Data Logging.

Downloading Firmware to the Controller

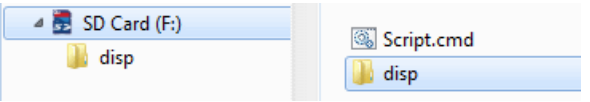
This table describes how to download a firmware to the logic controller using an SD card:

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 `script.cmd` in the SD card root directory.
4	Edit the file and insert the following command: `Download "/sys/os"`
5	Create the folder path `\sys\os` in the SD card root directory and copy the firmware file in the `os` folder:  **NOTE:** A firmware file example and the script are available in the folder `Firmwares & PostConfiguration\M221\` of the SoMachine Basic installation folder. The firmware file name for the M221 Logic Controller is `M221.mfw`.
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. **Result:** Copying of the firmware file begins. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress.
8	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` 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	Apply power to the logic controller.
2	Connect the Remote Graphic Display to the logic controller *(see Modicon TMH2GDB, Remote Graphic Display, User Guide)*.
3	Insert an empty SD card into the PC that is running SoMachine Basic.
4	Create a file called `script.cmd` in the SD card root directory.
5	Edit the file and insert the following command: `Download "/disp/TMH2GDB.mfw"`
6	Create the folder path `/disp/` in the SD card root directory and copy the firmware file in the `disp` folder:  **NOTE:** The firmware file and an example script are available in the folder `Firmwares & PostConfiguration\TMH2GDB\` of the SoMachine Basic installation folder. The firmware file name for the Remote Graphic Display is `TMH2GDB.mfw`.
7	Remove the SD card from the PC and insert it into the SD card slot of the M221 Logic Controller. **Result:** The logic controller begins transferring the firmware file from the SD card to the Remote Graphic Display. During this operation: • the message **File Transfer** is displayed on the Remote Graphic Display • the **SD** system LED on the M221 Logic Controller is illuminated • system word `%SW182` is set to 5 (**Display firmware transfer in progress)** **NOTE:** Do not disconnect the Remote Graphic Display or remove power from the M221 Logic Controller while the operation is in progress. The firmware update takes 5 to 6 minutes.
8	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` file. **NOTE:** Restoring the file system on the Remote Graphic Display (red backlight) is part of the process.

Downloading Firmware to TM3 Analog Expansion Modules

The firmware can be updated in TM3 analog expansion modules that have a firmware version greater than or equal to 26. If necessary, the version of firmware can be confirmed using SoMachine Basic.

Firmware updates are performed using a script file on an SD card. When the SD card is inserted in the SD card slot of the M221 Logic Controller, the logic controller updates the firmware of the TM3 analog expansion modules on the I/O bus, including those that are:

- Connected remotely, using a TM3 Transmitter/Receiver module
- In configurations comprising a mix of TM3 and TM2 expansion modules.

This table describes how to download a firmware to one or more TM3 analog expansion modules using an SD card:

Step	Action
1	Apply power to the logic controller.

2	Ensure that the logic controller is in the `EMPTY` state by deleting the application in the logic controller. You can do this with SoMachine Basic by using one of the following script commands:
	`Delete "usr/*"`
	`Delete "usr/app"`
	Refer to File Management Operations *(see page 198)* for details.

3	Insert an empty SD card into the PC.
4	Create a file called `script.cmd` in the SD card root directory.

5	Edit the file and insert the following command:
	`Download "/TM3/<filename>/*"`
	NOTE: `<filename>` is the file name of the firmware you wish to update. The asterisk signifies that all analog modules will be updated.
	To download the firmware to one specific TM3 analog expansion module, replace the asterisk with the position of the expansion module in the configuration. For example, to specify the module at position 4:
	`Download "/TM3/<filename>/4"`

6	Create the folder path `/TM3/` in the SD card root directory and copy the firmware file to the `TM3` folder. **NOTE:** A firmware file (the firmware file valid at the time of the installation of SoMachine Basic) and an example script are available in the folder `Firmwares & PostConfiguration\TM3\` of the SoMachine Basic installation folder.

Step	Action
7	Remove the SD card from the PC and insert it into the SD card slot of the M221 Logic Controller. **Result:** The logic controller begins transferring the firmware file from the SD card to the updatable TM3 analog expansion modules or to the one module specified in step 5. During this operation, the **SD** system LED on the M221 Logic Controller is illuminated. **NOTE:** The firmware update takes 10 to 15 seconds for each expansion module being updated. Do not remove power from the M221 Logic Controller, or remove the SD card, while the operation is in progress. Otherwise, the firmware update may be unsuccessful and the modules may no longer function correctly. In this case, run the Recovery Procedure *(see Modicon TM3 (SoMachine Basic), Expansion Modules Configuration, Programming Guide)* to reinitialize the firmware on the modules.
8	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` file.

If you remove power to the device, or there is a power outage or communication interruption during the transfer of the application, your device may become inoperative. If a communication interruption or a power outage occurs, reattempt the transfer. If there is a power outage or communication interruption during a firmware update, or if an invalid firmware is used, your device will become inoperative. In this case, use a valid firmware and reattempt the firmware update.

NOTICE

INOPERABLE EQUIPMENT

- Do not interrupt the transfer of the application program or a firmware change once the transfer has begun.
- Re-initiate the transfer if the transfer is interrupted for any reason.
- Do not attempt to place the device (logic controller, motion controller, HMI controller or drive) into service until the file transfer has completed successfully.

Failure to follow these instructions can result in equipment damage.

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.

To perform application management, the SD card name must be different from `DATA`, refer to Data Logging.

Backing Up an Application

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

Step	Action
1	Create a `script.cmd` file with a text editor on your PC.
2	Edit the file and insert the following line: `Upload "/usr/app"`
3	Copy the script file to the root folder of the SD card.
4	Insert the prepared SD card in the controller. **Result:** Copying of the application file begins. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress. **NOTE:** 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 `RUNNING` state compared to the `STOPPED` state.
5	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` file. **Result:** The application file (`*.smbk`) is saved on the SD card.

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 `script.cmd` file in the root folder of the SD card with a text editor.
2	Replace the content of the script by the following line: `Download "/usr/app"`
3	Remove power from the controller.
4	Insert the prepared SD card in the controller.

Step	Action
5	Restore power to the controller. **Result:** Copying of the application file begins. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress.
6	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` file.
7	Remove the SD card to restart the controller.

Post Configuration Management

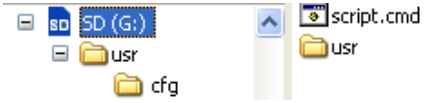
Overview

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

To perform post configuration management, the SD card name must be different from `DATA`, refer to Data Logging.

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 `script.cmd`.
2	Edit the file and insert the following line: `Download "/usr/cfg"`
3	Copy the post configuration file (`Machine.cfg`) to the folder `\usr\cfg` and the script file to the root folder of the SD card:  **NOTE:** A post configuration file example and the associated script are available in the directory `Firmwares & PostConfiguration\PostConfiguration\add_change\` of the SoMachine Basic installation directory.
4	If necessary, edit the `Machine.cfg` file to configure your post configuration parameters.
5	Insert the prepared SD card in the controller. **Result:** Downloading of the post configuration file begins. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress. **NOTE:** 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. **NOTE:** If a post configuration parameter is incompatible with the physical configuration, it is ignored.
6	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` 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 `script.cmd` file with a text editor on your PC.
2	Edit the file and insert the following line: `Upload "/usr/cfg"`
3	Copy the script file to the root folder of the SD card.
4	Insert the prepared SD card in the controller. **Result:** Copying of the post configuration file begins. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress. **NOTE:** 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 `RUNNING` state compared to the `STOPPED` state.
5	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` file. **Result:** 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 `script.cmd`.
3	Edit the file and insert the following line: `Delete "/usr/cfg"`
4	Copy the script file available in the directory `Firmwares & PostConfiguration\PostConfiguration\remove\` of the SoMachine Basic installation directory to the root directory of the SD card.
5	Insert the prepared SD card in the controller. **Result:** The post configuration file is removed. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress.
6	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` 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.

To perform error log management, the SD card name must be different from `DATA`, refer to Data Logging.

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 `script.cmd` file with a text editor on your PC.
2	Edit the file and insert the following line: `Upload "/sys/log"`
3	Copy the script file to the root folder of the SD card.
4	Insert the prepared SD card in the logic controller. **Result:** Transfer of the error log file begins. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress.
5	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` file. **Result:** The error log file (`PlcLog.csv`) 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 `script.cmd` file with a text editor on your PC.
2	Edit the file and insert the following line: `Delete "/sys/log"`
3	Copy the script file to the root folder of the SD card.
4	Insert the prepared SD card in the logic controller. **Result:** Deleting of the error log file begins. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress.
5	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` file. **Result:** The error log file (`PlcLog.csv`) 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 the following parts:

- Date and time
- Level
- Context
- Error code
- Priority (internal use only)

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

```
02/06/14, 12:04:01, 0x0111000100
```

This table describes the meaning of the hexadecimal error representation:

Group	Error code (hex)	Error Description	Result
General	08000011xx	Invalid hardware calibration parameters	Ethernet channel is inoperative %SW118.bit10 set to 0 **ERR** LED flashes
Operating system	0F01xxxxxx	Operating system error detected	Transition to HALTED state
Memory management	0F030009xx	Internal memory allocation error detected	Transition to HALTED state
SD card	010C001Bxx	Error while accessing an SD card; the operation exceeded an internal timeout (3000 ms).	The SD card operation is canceled.
Watchdog timer	0104000Axx	Logic controller resource utilization greater than 80% - first detection	Watchdog timeout signaled: %S11 set to 1 **ERR** LED flashes
	0804000Bxx	Logic controller resource utilization greater than 80% - second consecutive detection	Transition to HALTED state
	0804000Cxx	Task watchdog timer in master task	Transition to HALTED state
	0804000Dxx	Task watchdog timer in periodic task	Transition to HALTED state
Battery	0105000Exx	Battery is depleted	Depleted battery signaled: %S75 set to 1 **BAT** LED illuminated
RTC	01060012xx	RTC is invalid	Invalid RTC signaled: %SW118.bit12 set to zero %S51 set to 1
User application	0807000Fxx	Application is not compatible with firmware	Transition to EMPTY state
	08070010xx	Checksum error detected	Transition to EMPTY state

Group	Error code (hex)	Error Description	Result
Ethernet	010B0014xx	Duplicate IP address detected	Duplicate IP signaled: %SW62 set to 1 %SW118.bit9 set to 0 **ERR** LED flashes
Embedded I/O	010D0013xx	Short-circuit detected on protected output	Over-current signaled: %SW139 set to 1 (depending on the output block) **ERR** LED flashes
Read non-volatile memory	01110000xx	Read error detected - file not found	Unsuccessful read operation
	01110001xx	Read error detected - incorrect logic controller type	
	01110002xx	Read error detected - incorrect header	
	01110003xx	Read error detected - incorrect area descriptor	
	01110004xx	Read error detected - incorrect area descriptor size	
Write non-volatile memory	01120002xx	Write error detected - incorrect header	Unsuccessful write operation
	01120004xx	Write error detected - incorrect area descriptor size	
	01120005xx	Write error detected - unsuccessful erase	
	01120006xx	Write error detected - incorrect header size	
Persistent variable	01130007xx	Checksum error detected in persistent variables	Persistent variables cannot be restored
	01130008xx	Size error detected in persistent variables	
Ethernet IP	01140012xx	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 `script.cmd` file with a text editor on your PC.
2	Edit the file and insert the following line: `Upload "/usr/mem"`
3	Copy the script file to the root folder of the SD card.
4	Insert the prepared SD card in the controller. **Result:** Copying of the memory begins. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress. **NOTE:** 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 `RUNNING` state compared to the `STOPPED` state.
5	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` file. **Result:** The memory file (`*.csv`) is saved on the SD card.

Restoring Controller Memory or Copying to Another Controller

Step	Action
1	Edit the `script.cmd` file in the root folder of the SD card with a text editor.
2	Replace the content of the script by the following line: `Download "/usr/mem"`
3	Insert the prepared SD card in the controller. **Result:** Copying of the memory file begins. During the operation, the **SD** system LED on the logic controller is illuminated. **NOTE:** Avoid removing power from the logic controller while the operation is in progress.
4	Wait until the end of the operation (until the **SD** LED is off or flashing). If an error is detected, the **SD** and **ERR** LEDs flash and the detected error is logged in `Script.log` 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 8
I/O Objects

What Is in This Chapter?

This chapter contains the following topics:

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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **I/O objects → Digital inputs**. **Result**: 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
Used	No	True/False	False	Indicates whether the input channel is being referenced in a program.
Address	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.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this input. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with this address. Double-click in the **Comment** 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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **I/O objects → Digital outputs**. **Result**: 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
Used	No	True/False	False	Indicates whether the output channel is being referenced in a program.
Address	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.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this output. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	The comment associated with this address. Double-click in the **Comment** 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 *(see Modicon M221 Logic Controller, 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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **I/O objects → Analog inputs**. **Result**: 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
Used	No	True/False	False	Indicates whether the input channel is being referenced in a program.
Address	No	%IW0.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, `%IW0.1` is the analog input at the analog input channel number 1 of the logic controller.
		%IW0.x0y	–	Displays the address of the analog output channel on the cartridge, where x is the cartridge number and and y is the channel number.

Parameter	Editable	Value	Default Value	Description
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this input. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	The comment associated with this address. Double-click in the **Comment** 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 recieved 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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **I/O objects → Analog outputs**. **Result**: 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
Used	No	True/False	False	Indicates whether the output channel is being referenced in a program.
Address	No	%QW0.x0y	–	Displays the address of the analog output channel on the cartridge, where x is the cartridge number and and y is the channel number.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this output. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	The comment associated with this address. Double-click in the **Comment** 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:

Input Assembly (EtherNet/IP) 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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **Network objects → Input assembly (EtherNet/IP)**. **Result**: The properties window appears.

Input Assembly Properties

This table describes each property of an Input assembly object:

Parameter	Editable	Value	Default Value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
Address	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 *(see page 44)*.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.

Parameter	Editable	Value	Default Value	Description
Fallback value	Yes	-32768...32767	0	Specify the value to apply to this object when the logic controller enters the `STOPPED` or an exception state. **NOTE:** If **Maintain values** fallback mode is configured, the object retains its 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 *(see SoMachine Basic, Operating Guide)*.
Comment	Yes	–	–	A comment associated with this object. Double-click in the **Comment** column and type an optional comment to associate with this object.

Output Assembly (EtherNet/IP) 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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **Network objects → Output assembly (EtherNet/IP)**. **Result**: The properties window appears.

Output Assembly Properties

This table describes each property of an Output assembly object:

Parameter	Editable	Value	Default value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
Address	No	%IWEi	–	The address of the Output assembly, where i is the instance identifier. For the maximum number of instances, refer to Maximum Number of Objects *(see page 44)*.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with this object. Double-click in the **Comment** column and type an optional comment to associate with this object.

Input Registers (Modbus TCP) Objects (%QWM)

Introduction

Input registers objects are the digital values of Modbus TCP 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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **Network objects → Input registers (Modbus TCP)**. **Result**: The properties window appears.

Input Registers Properties

This table describes each property of an Input registers object:

Parameter	Editable	Value	Default value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
Address	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 *(see page 44)*.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.

Parameter	Editable	Value	Default value	Description
Fallback value	Yes	-32768...32767	0	Specify the value to apply to this object when the logic controller enters the `STOPPED` or an exception state. **NOTE:** If **Maintain values** fallback mode is configured, the object retains its 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 *(see SoMachine Basic, Operating Guide)*.
Comment	Yes	–	–	A comment associated with this object. Double-click in the **Comment** column and type an optional comment to associate with this object.

Output Registers (Modbus TCP) Objects (%IWM)

Introduction

Output registers objects are the digital values of Modbus TCP 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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **Network objects → Output registers (Modbus TCP)**. **Result**: The properties window appears.

Output Registers Properties

This table describes each property of an Output registers object:

Parameter	Editable	Value	Default value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
Address	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 *(see page 44)*.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with this object. Double-click in the **Comment** column and type an optional comment to associate with this object.

Digital Input (IOScanner) Objects (%IN)

Introduction

Digital input (IOScanner) objects are the digital values received from Modbus Serial IOScanner or Modbus TCP IOScanner devices.

Displaying Digital inputs (IOScanner) Properties

Follow these steps to display properties of Digital inputs (IOScanner) objects:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **Network objects → Digital inputs (IOScanner)**. **Result**: The properties window appears.

Digital inputs (IOScanner) Properties

This table describes each property of an Digital inputs (IOScanner) object:

Parameter	Editable	Value	Default value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in the program.
Address	No	%IN(i+x).y.z)	–	The address of the object, where: ● i: index: ❍ 100 for SL1 ❍ 200 for SL2 ❍ 300 for ETH1(Modbus TCP IOScanner) ● x: device ID ● y: channel ID ● z: object instance identifier For the maximum number of instances, refer to Maximum Number of Objects *(see page 44)*.
Channel	No	Name of the configured channel.	-	The name of the channel being used to receive the data from the device.

Parameter	Editable	Value	Default value	Description
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with this object. Double-click in the **Comment** column and type an optional comment to associate with this object.

Digital Output (IOScanner) Objects (%QN)

Introduction

Digital output (IOScanner) objects are the digital values sent to Modbus Serial IOScanner or Modbus TCP IOScanner devices.

Displaying Digital outputs (IOScanner) Properties

Follow these steps to display properties of Digital outputs (IOScanner) objects:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **Network objects → Digital outputs (IOScanner)**. **Result**: The properties window appears.

Digital outputs (IOScanner) Object Properties

This table describes each property of a Digital outputs (IOScanner) object:

Parameter	Editable	Value	Default value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
Address	No	%QN(i+x).y.z	–	The address of the object, where: ● i: index: ❍ 100 for SL1 ❍ 200 for SL2 ❍ 300 for ETH1(Modbus TCP IOScanner) ● x: device ID ● y: channel ID ● z: object instance identifier For the maximum number of instances, refer to Maximum Number of Objects *(see page 44)*.
Channel	Yes	Name of the configured channel.	-	The name of the channel being used to send the data to the device.
Fallback value	Yes	0 or 1	0	Specify the value to apply to this object when the logic controller enters the `STOPPED` or an exception state. **NOTE:** If **Maintain values** fallback mode is configured, the object retains its 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.

Parameter	Editable	Value	Default value	Description
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with this object. Double-click in the **Comment** column and type an optional comment to associate with this object.

Input Register (IOScanner) Objects (%IWN)

Introduction

Input register (IOScanner) objects are the register values received from Modbus Serial IOScanner or Modbus TCP IOScanner devices.

Displaying Input registers (IOScanner) Properties

Follow these steps to display properties of Input registers (IOScanner) objects:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **Network objects → Input registers (IOScanner)**. **Result**: The properties window appears.

Input registers (IOScanner) Properties

This table describes each property of an Input registers (IOScanner) object:

Parameter	Editable	Value	Default value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in the program.
Address	No	%IWN(i+x).y.z	–	The address of the object, where: ● i: index: ❍ 100 for SL1 ❍ 200 for SL2 ❍ 300 for ETH1(Modbus TCP IOScanner) ● x: device ID ● y: channel ID ● z: object instance identifier For the maximum number of instances, refer to Maximum Number of Objects *(see page 44)*.
Channel	No	Name of the configured channel.	-	The name of the channel being used to receive the data from the device.

Parameter	Editable	Value	Default value	Description
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with this object. Double-click in the **Comment** column and type an optional comment to associate with this object.

Output Register (IOScanner) Objects (%QWN)

Introduction

Output register (IOScanner) objects are the register values sent to Modbus Serial IOScanner or Modbus TCP IOScanner devices.

Displaying Output registers (IOScanner) Properties

Follow these steps to display properties of Output registers (IOScanner) objects:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **Network objects → Output registers (IOScanner)**. **Result**: The properties window appears.

Output registers (IOScanner) Object Properties

This table describes each property of a Output registers (IOScanner) object:

Parameter	Editable	Value	Default value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the object is being referenced in a program.
Address	No	%QWN(i+x).y.z	–	The address of the object, where: ● i: index: ❍ 100 for SL1 ❍ 200 for SL2 ❍ 300 for ETH1(Modbus TCP IOScanner) ● x: device ID ● y: channel ID ● z: object instance identifier For the maximum number of objects, refer to Maximum Number of Objects *(see page 44)*.
Channel	Yes	Name of the configured channel.	-	The name of the channel being used to send the data to the device.

Parameter	Editable	Value	Default value	Description
Fallback value	Yes	-32768...32767	0	Specify the value to apply to this object when the logic controller enters the `STOPPED` or an exception state. **NOTE:** If **Maintain values** fallback mode is configured, the object retains its 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 *(see SoMachine Basic, Operating Guide)*.
Symbol	Yes	–	–	The symbol associated with this address. Double-click in the **Symbol** column and type the name of the symbol to associate with this object. If a symbol already exists, you can right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of this symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with this object. Double-click in the **Comment** column and type an optional comment to associate with this object.

Modbus IOScanner Network Diagnostic Codes (%IWNS)

Device Diagnostic Codes

The following table shows the possible values of the diagnostic codes returned by device x in the corresponding Modbus IOScanner network diagnostic object (`%IWNS(100+x)` for SL1, or `%IWNS(200+x)` for SL2, `%IWNS(300+x)` for ETH1):

Value	Description
0	Device not scanned.
1	Device is being initialized by Modbus IOScanner (Initialization request of device being sent).
2	Device is present and ready to be scanned (initialization requests sent, if any).
3	Device not scanned correctly due to a communication error detected on a channel of the device.
4	Device not initialized correctly due to a communication error detected during initialization request of the device.
5	Device not correctly identified because the vendor name or product code returned by the device does not match the expected values.
6	Communication error occurred during identification and initialization. Possible reasons are: incommunicative or absent device, incorrect communication parameters, or unsupported Modbus function.

Channel Diagnostic Codes

The following table shows the possible values of the diagnostic codes returned by device x and channel y in the corresponding Modbus IOScanner network diagnostic object (`%IWNS(100+x).y` for SL1, `%IWNS(200+x).y` for SL2, `%IWNS(300+x)` for ETH1):

Value	Description
0	Channel is active
-1	Channel is inactive
Other	Value of the Communication error code (CommError) *(see SoMachine Basic, Generic Functions Library Guide)*

Chapter 10
System Objects

What Is in This Chapter?

This chapter contains the following topics:

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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **System objects → System Bits**. **Result**: System bit properties appear on the screen.

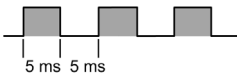
System Bits Properties

This table describes each property of the system bit:

Parameter	Editable	Value	Default Value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the system bit is being referenced in a program.
Address	No	%Si	–	Displays the system bit address, where i is the bit number that represents the sequential position of the system bit in the memory. If the controller has maximum n system bits, the value of i is given as 0...n-1. For example, `%S4` is system bit 4.
Symbol	Yes	–	–	The symbol associated with the system bit. Double-click in the **Symbol** 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 **Symbol** column and choose **Search and Replace** to find and replace occurrences of the symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with the system bit. Double-click in the **Comment** 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: • A power return with loss of data (battery malfunction), • The program or an animation table. 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)
%S9	Fallback outputs	When %S9 is set to 1: • For outputs configured as Status Alarms, PTO, or FREQGEN, the outputs are set to 0. • Fallback values are applied to the physical digital and analog outputs (embedded outputs, TM2/TM3 expansion module outputs and TMC2 cartridge outputs). The data image is unaffected by %S9. The data image reflects the logic applied by the application. Only the physical outputs are affected. • The fallback values are applied regardless of the fallback behavior *(see SoMachine Basic, Operating Guide)* mode configured for specific outputs. When %S9 is set to 0, the data image values are re-applied to the physical outputs. **NOTE:** When the controller is in the STOPPED state and **Maintain values** fallback behavior is configured, a rising edge on %S9 applies fallback values to the physical outputs and to data image values.	0	U

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%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 **ERR** 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 254)*). 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: ● 1 when the controller state is RUNNING, ● 0 for STOPPED, BOOTING, or any other state.	0	S, SIM
%S13	First cycle in RUNNING state	Normally set to 0. Set to 1 by the system during the first scan after the controller state has been changed to RUNNING.	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

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%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: ● A result greater than + 32767 or less than - 32768, in single length, ● A result greater than + 2147483647 or less than - 2147483648, in double length, ● A result greater than + 3.402824E+38 or less than - 3.402824E+38, in floating point, ● Division by 0, ● The square root of a negative number, ● BTI or ITB conversion not significant: BCD value out of limits. 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
%S21	Grafcet initialization	Normally set to 0, it is set to 1 by: ● A cold restart, %S0 = 1, ● The program, in the preprocessing program part only, using a Set Instruction (S %S21) or a set coil –(S)– %S21, ● The terminal. 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.	0	U→S, SIM

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S22	Grafcet reset	Normally set to 0, it can only be set to 1 by the program in pre-processing. 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.	0	U→S, SIM
%S23	Preset and freeze Grafcet (List)	Normally set to 0, it can only be set to 1 by the program in the pre-processing program module. Set to 1, it validates the pre-positioning of Grafcet (List). Maintaining this bit at 1 freezes Grafcet (List) execution. It is reset to 0 by the system at the start of the execution of the sequential processing.	0	U→S, SIM
%S28	String overflow	Set to 1, it indicates that there is an overflow in a memory object when managing strings.	0	S→U, SIM
%S33	Read or Write selection for Ethernet server configuration read/change	Normally set to 0. ● 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). ● Set to 1 (if there is no post configuration in use), then the new configuration is given by %SW33 to %SW38. 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.	0	U→S
%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. **NOTE:** A modification in the state of %S34, %S35, or %S36 provokes a reinitialization of the Ethernet channel. As a consequence, the Ethernet channel may not be available for several seconds after the modification.	0	U

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%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: ● Set to 0 if Half Duplex, ● Set to 1 if Full Duplex. **NOTE:** A modification in the state of %S34, %S35, or %S36 provokes a reinitialization of the Ethernet channel. As a consequence, the Ethernet channel may not be available for several seconds after the modification.	–	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: ● Set to 0 if 10 Mbps, ● Set to 1 if 100 Mbps. **NOTE:** A modification in the state of %S34, %S35, or %S36 provokes a reinitialization of the Ethernet channel. As a consequence, the Ethernet channel may not be available for several seconds after the modification.	–	U or S
%S38	Permission for events to be placed in the events queue	Normally set to 1. ● Set to 0, events cannot be placed in the events queue. ● Set to 1, events are placed in the events queue as soon as they are detected, This bit can be set to its initial state 1 by the program and the system (on cold restart).	1	U→S
%S39	Saturation of the events queue	Normally set to 0. ● Set to 0, all events are reported. ● Set to 1, at least one event is lost. This bit can be set to 0 by the program and the system (on cold restart).	0	U→S

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S49	Output rearming *(see page 65)*	Normally set to 0, this bit can be set to 1 or 0 by the program. ● Set to 0, the automatic re-arming of outputs following a short circuit is disabled. ● Set to 1, the automatic re-arming of outputs following a short circuit is enabled. **NOTE:** The bit is reset to 0 on a cold start; otherwise, the bit value is retained. 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. **NOTE:** %S10 and %SW139 are reset to their initial state when %S49 is set to 1.	0	U→S

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S50	Updating the date and time using words %SW49 to %SW53	Normally set to 0, this bit can be set to 1 or 0 by the program. ● Set to 0, the date and time can be read. ● Set to 1, the date and time can be updated but not read. While %S50 is set to 1, the controller date and time are no longer updated by the system and cannot be read by the user program. The internal RTC controller is updated on a falling edge of %S50. Process details: ● If %S50=0, the controller regularly updates the system words %SW49-53 from its internal clock. Reading %SW49-53 then provides the controller internal date & time. ● Setting %S50 to 1 stops this update and allows to write to %SW49-53 without being overwritten by the above process. ● When the controller detects a falling edge of %S50 (from 1 to 0), it applies the values of %SW49-53 to its internal clock and restarts the update of %SW49-53. This %S50 process is also the mechanism used by SoMachine Basic to update the controller time from the RTC management view. So if SoMachine Basic detects that %S50 is already set to 1, a message informs that SoMachine Basic cannot read the exact value of the controller internal clock. However, this situation does not prevent updates to the controller date & time from the RTC management view but, if used, %S50 will be reset by SoMachine Basic.	0	U→S

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S51	Time-of-day clock status	Normally set to 0, this bit can be set to 1 or 0 by the program. ● Set to 0, the date and time are consistent. ● Set to 1, the date and time must be initialized by the program. 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). State 1 transitioning to state 0 forces a Write of the correction constant to the RTC.	0	U→S, SIM
%S52	RTC write error detected	This bit, managed by the system, is set to 1 to indicate that an RTC write (requested by %S50) has not been performed because of invalid values in %SW49 to %SW53 *(see page 255)*. This bit is set to 0 if the requested RTC change has been correctly applied.	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. ● Set to 0, the system word %SW59 is not managed, ● Set to 1, the date and time are incremented or decremented according to the rising edges on the control bits set in %SW59.	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: ● Set to 0, the external battery is operating normally. ● Set to 1, external battery power is low, or no external battery is detected.	0	S
%S90	Backup/Restore/Erase destination	This system bit selects the destination of the memory words backup/restore/erase operation: ● Set to 0: non-volatile memory (default). ● Set to 1: SD card.	0	U
%S91	Erase backed up variables	Set this bit to 1 to erase the backed up variables stored in non-volatile memory or the SD card, depending on %90.	–	U→S
%S92	%MW variables backed up in non-volatile memory	This system bit is set to 1 by the system if memory word (%MW) variables are available in non-volatile memory.	–	S

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S93	Back up %MW	Set this bit to 1 to back up the %MW variables in the non-volatile memory or SD card, depending on %S90.	–	U→S
%S94	Restore **%MW**	Set this bit to 1 to restore the data backed up in non-volatile memory or SD card, depending on %S90.	–	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. ● Set to 0, the backup program is invalid. ● Set to 1, the backup program is valid.	0	S, SIM
%S101	Changing a port address (Modbus protocol)	Used to change a serial line port address using system words %SW101 (SL1) and %SW102 (SL2). To do this, %S101 must be set to 1. ● Set to 0, the address cannot be changed. The value of %SW101 and %SW102 matches the current port address, ● Set to 1, the address can be changed by changing the values of %SW101 (SL1) and %SW102 (SL2). **NOTE:** %S101 cannot be set to 1 if a post configuration file is defined on SL1 or SL2.	0	U

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

System Bit	Function	Description	Init State	Control
%S103 %S104	Using the ASCII protocol	Enables the use of the ASCII protocol on SL1 (%S103) or SL2 (%S104). The ASCII protocol is configured using system words %SW103 and %SW105 for SL1, and system words %SW104 and %SW106 for SL2. ● Set to 0, the protocol used is the one configured in SoMachine Basic, or specified in the post configuration *(see page 70)*. ● Set to 1, the ASCII protocol is used on SL1 (%S103) or SL2 (%S104). In this case, the system words %SW103, %SW105, and %SW121 must be previously configured for SL1, and %SW104, %SW106, and %SW122 for SL2. Each change of those %SW will be taken into account after a rising edge to %S103 or %S104. **NOTE:** A rising or falling edge on %S103 or %S104 cancels an exchange in progress (EXCH instruction). **NOTE:** Setting %S103 or %S104 to 0 reconfigures the serial line with the SoMachine Basic parameters. **NOTE:** %S103 and %S104 are ignored if a Modbus Serial Line IOScanner is configured on the corresponding serial line.	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 *(see page 254)*.	0	U/S
%S106	I/O bus behavior	The default value is 0, meaning that a bus communication error on an expansion module *(see page 124)* stops the I/O expansion bus exchanges. Set this bit to 1 to specify that the controller continues to make I/O expansion bus exchanges. **NOTE:** When a bus communication error occurs, bit n of %SW120 is set to 1, where n is the expansion module number, and %SW118 bit 14 is set to 0. For more information on bus error handling, refer to I/O Configuration General Description *(see page 124)*.	0	U/S
S Controlled by the system **U** Controlled by the user **U→S** Set to 1 by the user, reset to 0 by the system **S→U** Set to 1 by the system, reset to 0 by the user **SIM** Applied in the Simulator				

System Bit	Function	Description	Init State	Control
%S107	I/O bus restart	The default value is 0. Reset to 0 by the system. Set this bit to 1 to force a restart of the I/O expansion bus *(see page 126)*. On detection of a rising edge of this bit, the logic controller reconfigures and restarts the I/O expansion bus if: ● %S106 is set to 0 (that is, I/O exchanges are stopped) ● %SW118 bit 14 is set to 0 (I/O bus is in error) ● At least one bit of %SW120 is set to 1 (identifying the module that is in bus communication error) For more information on bus error handling, refer to I/O Configuration General Description *(see page 124)*.	0	U/S
%S110	IOScanner reset SL1	Set to 1 to reset the Modbus Serial IOScanner on Serial Line 1.	0	U/S
%S111	IOScanner reset SL2	Set to 1 to reset the Modbus Serial IOScanner on Serial Line 2.	0	U/S
%S112	IOScanner reset ETH1	Set to 1 to reset the Modbus TCP IOScanner on Ethernet.	0	U/S
%S113	IOScanner suspend SL1	Set to 1 to suspend the Modbus Serial IOScanner on Serial Line 1.	0	U/S
%S114	IOScanner suspend SL2	Set to 1 to suspend the Modbus Serial IOScanner on Serial Line 2.	0	U/S
%S115	IOScanner suspend ETH1	Set to 1 to suspend the Modbus TCP IOScanner on Ethernet.	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 **Alarm** 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

S Controlled by the system
U Controlled by the user
U→S Set to 1 by the user, reset to 0 by the system
S→U Set to 1 by the system, reset to 0 by the user
SIM Applied in the Simulator

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 **Tools** tab in the left-hand area of the **Programming** window.
2	Click **System objects → System Words**. **Result**: 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
Used	No	TRUE/FALSE	FALSE	Indicates whether the system word is being referenced in a program.
Address	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.
Symbol	Yes	–	–	The symbol associated with the system word. Double-click in the **Symbol** 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 **Symbol** column and choose **Search and Replace** to find and replace occurrences of the symbol throughout the program and/or program comments.
Comment	Yes	–	–	A comment associated with the system word. Double-click in the **Comment** 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 the controller scan period (1...150 ms) defined in the Master task properties *(see SoMachine Basic, Operating Guide)* or an animation table.	U, SIM
%SW1	Periodic task period	Modifies the cycle time [1...255 ms] of the periodic task, without losing the **Period** value specified in the periodic task properties window. Allows you to recover the **Period** value saved in the periodic task properties window: • in case of a cold start, or • if the value you write in %SW1 is outside [1...255] range. 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
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%SW7	Controller status	• Bit [0]: Backup/restore in progress: ❍ Set to 1 if backup/restore of the program is in progress, ❍ Set to 0 if backup/restore of the program is complete or disabled. • Bit [1]: Configuration of the controller is OK: ❍ Set to 1 if configuration ok. • Bit [2]: SD card status bits: ❍ Set to 1 if SD card is present. • Bit [3]: SD card status bits: ❍ Set to 1 if SD card is being accessed. • Bit [4]: Application memory status: ❍ Set to 1 if application in RAM memory is different from that in non-volatile memory. • Bit [5]: SD card status bits: ❍ Set to 1 if SD card is in error. • Bit [6]: Not used (status 0) • Bit [7]: Controller reserved: ❍ Set to 1 when the controller is connected to SoMachine Basic. • Bit [8]: Application in Write mode: ❍ Set to 1 if application is protected. In this case, the clone operation does not replicate the application (refer to Clone Management *(see page 202)*). • Bit [9]: Not used (status 0) • Bit [10]: Second serial port installed as cartridge (compact only): ❍ 0 = No serial cartridge ❍ 1 = Serial cartridge installed • Bit [11]: Second serial port type: ❍ Set to 1 = EIA RS-485 • Bit [12]: Validity of the application in internal memory: ❍ Set to 1 if the application is valid. • Bit [14]: Validity of the application in RAM memory: ❍ Set to 1 if the application is valid. • Bit [15]: Ready for execution: ❍ Set to 1 if ready for execution.	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
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%SW13	Boot loader version xx.yy	For example, if %SW13=000E hex: • 8 MSB=00 in hexadecimal, then xx=0 in decimal • 8 LSB=0E in hexadecimal, then yy=14 in decimal As a result, boot loader version is 0.14, displayed as 14 decimal.	S, SIM
%SW14	Commercial version, xx.yy	For example, if %SW14=0232 hex: • 8 MSB=02 in hexadecimal, then xx=2 in decimal • 8 LSB=32 in hexadecimal, then yy=50 in decimal 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: • 8 MSB=00 in hexadecimal, then aa=00 in decimal • 8 LSB=03 in hexadecimal, then bb=03 in decimal %SW16=0B16 hex: • 8 MSB=0B in hexadecimal, then cc=11 in decimal • 8 LSB=16 in hexadecimal, then dd=22 in decimal 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: • Bit[0]: Invalid operation, result is not a number (NaN) • Bit[1]: Reserved • Bit[2]: Division by 0, result is invalid (-Infinity or +Infinity) • Bit[3]: Result greater in absolute value than +3.402824e+38, result is invalid (-Infinity or +Infinity).	S and U, SIM
%SW18-%SW19	100 ms absolute timer counter	This counter works using 2 words: • %SW18 represents the least significant word, • %SW19 represents the most significant word. The double word (%SW18-%SW19) increases from 0 to 2^31 each 100 ms as a counter modulo 2^31.This double word is also reset during the initialization 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). **NOTE:** 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
S Controlled by the system **U** Controlled by the user **SIM** 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. **NOTE:** 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: ● T_{ON} ≥ 1 ms ● 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): P ≥ 2 x %SW31.	S
%SW32	Min. scan time (master task)	Indicates the execution time of shortest controller scan cycle since the last cold start (in ms). **NOTE:** 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: ● IP address: %SW33 and %SW34 For IP address AA.BB.CC.DD: %SW33 = CC.DD and %SW34 = AA.BB ● Subnetwork mask: %SW35 and %SW36 For subnetwork mask AA.BB.CC.DD: %SW35 = CC.DD and %SW36 = AA.BB ● Gateway address: %SW37 and %SW38 For gateway address AA.BB.CC.DD: %SW37 = CC.DD and %SW38 = AA.BB	U
%SW39	Periodic average time	Indicates the average execution time in µs of the periodic task (last 5 times).	–
%SW40	Event 0 average time	Indicates the average execution time in µs of the event task associated with the input %I0.2 (last 5 times).	–
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description		Control
%SW41	Event 1 average time	Indicates the average execution time in µs of the event task associated with the input %I0.3 (last 5 times).		–
%SW42	Event 2 average time	Indicates the average execution time in µs of the event task associated with the input %I0.4 (last 5 times).		–
%SW43	Event 3 average time	Indicates the average execution time in µs of the event task associated with the input %I0.5 (last 5 times).		–
%SW44	Event 4 average time	Indicates the average execution time in µs 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 µs 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 µs 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 µs 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.) **NOTE:** 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):		S and U, SIM
		%SW49	xN Day of the week (N=1 for Monday) **NOTE:** %SW49 is read only (S).	
		%SW50	00SS Seconds	
		%SW51	HHMM: hour and minute	
		%SW52	MMDD: month and day	
		%SW53	CCYY: century and year	
		Set system bit %S50 to 1 to enable updating the RTC value using system words %SW49 to %SW53. On a falling edge of %S50 the internal RTC controller is updated using the values written in these words. For more details, see system bit %S50 *(see page 243)*.		

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description		Control
%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):		S, SIM
		%SW54	SS Seconds	
		%SW55	HHMM: hour and minute	
		%SW56	MMDD: month and day	
		%SW57	CCYY: century and year	

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description		Control
%SW58	Code of last stop	Displays code giving cause of last transition from the RUNNING state to another state:		S, SIM
		0	Initial value (after a download or an initialization command)	
		1	Run/Stop input or Run/Stop switch is set to 0. A falling edge on the Run/Stop input or Run/Stop switch at 0 has been detected while the controller was in the RUNNING state, or the controller was powered on with the Run/Stop input or Run/Stop switch at 0.	
		2	Program error detected. A program error has been detected while the controller was in the RUNNING state (in which case the controller goes to the HALTED state), or the controller was in the HALTED state when the power was cycled, preventing it starting in run.	
		3	Stop command using SoMachine Basic online button or Remote Graphic Display.	
		4	Power outage. The controller starting in run after a power cycle, or the controller is in the STOPPED state because the starting mode is **Start in Previous State** and the controller was in the STOPPED state when the power outage occurred.	
		5	Hardware error detected.	
		6	Not used.	
		7	Power on with starting mode configured as **Start in stop**.	
		8	The controller could not recover previous data that it had at last power outage (for example because the battery is low), preventing it starting in run.	
		9	Controller is not able to run due to internal memory errors.	
		The reasons for the last stop are prioritized in the following order (that is, when the controller is in the STOPPED state after a power cycle): 1, 7, 4, 8, 2		

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

<table>
<tr><th>System Words</th><th>Function</th><th>Description</th><th>Control</th></tr>
<tr><td>%SW59</td><td>Adjust current date</td><td>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.
<table>
<tr><th>Increment</th><th>Decrement</th><th>Parameter</th></tr>
<tr><td>bit 0</td><td>bit 8</td><td>Day of week</td></tr>
<tr><td>bit 1</td><td>bit 9</td><td>Seconds</td></tr>
<tr><td>bit 2</td><td>bit 10</td><td>Minutes</td></tr>
<tr><td>bit 3</td><td>bit 11</td><td>Hours</td></tr>
<tr><td>bit 4</td><td>bit 12</td><td>Days</td></tr>
<tr><td>bit 5</td><td>bit 13</td><td>Month</td></tr>
<tr><td>bit 6</td><td>bit 14</td><td>Years</td></tr>
<tr><td>bit 7</td><td>bit 15</td><td>Centuries</td></tr>
</table></td><td>U</td></tr>
<tr><td>%SW62</td><td>Ethernet error detection</td><td>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</td><td>S</td></tr>
<tr><td colspan="4">S Controlled by the system
U Controlled by the user
SIM Applied in the simulator</td></tr>
</table>

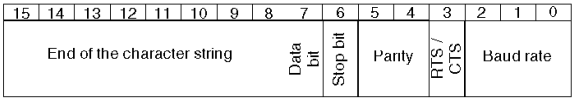
System Words	Function	Description	Control
%SW63	EXCH1 block error code	EXCH1 error code: 0 - operation was successful 1 - number of bytes to be transmitted exceeds the limit (> 255) 2 - insufficient transmission table 3 - insufficient word table 4 - receive table overflowed 5 - time-out elapsed 6 - transmission 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) 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 *(see page 278)*.	S, SIM

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

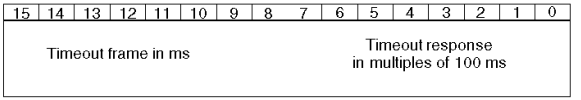
System Words	Function	Description	Control
%SW70	Scan time microseconds resolution	Indicates the execution time of the last controller scan cycle (in µs). **NOTE:** This time corresponds to the time elapsed between the start (acquisition of inputs) and the end (update of outputs) of a master task scan 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). **NOTE:** 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). **NOTE:** 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
%SW80	Status of embedded analog inputs	• Bit [0]: Set to 1 if the embedded analog inputs are operational • Bit [6]: Set to 1 if an error is detected on analog input 0 • Bit [7]: Set to 1 if an error is detected on analog input 1 • All other bits are reserved and set to 1	S and U, SIM
%SW94 %SW95	Application signature %MW60028–%MW60034	If the application changes, in terms of configuration or programming data, the signature (sum of all checksums) also changes. If %SW94 = 91F3 in hexadecimal, the application signature is 91F3 in hexadecimal.	S, SIM
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%SW96	Diagnostics for save/restore function of program and %MW	• Bit [1]: This bit is set by the firmware to indicate when the save is complete: ❍ Set to 1 if the backup is complete. ❍ Set to 0 if a new backup is requested. • Bit [2]: Back up error detected, refer to bits 8, 9, 10, 12 and 14 for further information: ❍ Set to 1 if an error is detected. ❍ Set to 0 if a new backup is requested. • Bit [6]: Set to 1 if the controller contains a valid application in RAM memory. • Bit [10]: Difference detected between RAM memory and non-volatile memory. ❍ Set to 1 if there is a difference. • Bit [12]: Indicates if a restore error has occurred: ❍ Set to 1 if an error is detected. • Bit [14]: Indicates if a non-volatile memory write error has occurred: ❍ Set to 1 if an error is detected.	S, SIM
%SW98	Post configuration status (Serial Line 1)	The bits are set to 1 when the post configuration was applied for the parameter: • Bit[0]: Hardware option (RS485 or RS232) • Bit[1]: Baudrate • Bit[2]: Parity • Bit[3]: Data size • Bit[4]: Number of stop bits • Bit[5]: Modbus address • Bit[6]: Polarization (if available in the port)	S
%SW99	Post configuration status (Serial Line 2)	The bits are set to 1 when the post configuration was applied for the parameter: • Bit[0]: Hardware option (RS485) • Bit[1]: Baudrate • Bit[2]: Parity • Bit[3]: Data size • Bit[4]: Number of stop bits • Bit[5]: Modbus address • Bit[6]: Polarization (if available in the port)	S
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%SW100	Post configuration status (Ethernet)	The bits are set to 1 when the post configuration was applied for the parameter: • Bit[0]: IP mode (fixed, DHCP, or BOOTP) • Bit[1]: IP address • Bit[2]: Network submask • Bit[3]: Default gateway • Bit[4]: Device name **NOTE:** 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.	S
%SW101 %SW102	Value of the Modbus address port	When bit %S101 is set to 1, you can change the Modbus address of SL1 or SL2. The address of SL1 is %SW101. The address of SL2 is %SW102. **NOTE:** The update is applied immediately after writing a new address to %SW101 or %SW102.	U
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%SW103 %SW104	Configuration for use of the ASCII protocol	When bit %S103 (SL1) or %S104 (SL2) is set to 1, the ASCII protocol is used. System word %SW103 (SL1) or %SW104 (SL2) must be set according to the elements below:  • Baud rate: ❍ 000: 1200 baud, ❍ 001: 2400 baud, ❍ 010: 4800 baud, ❍ 011: 9600 baud, ❍ 100: 19200 baud, ❍ 101: 38400 baud, ❍ 110: 57600 baud, ❍ 111: 115200 baud. • RTS/CTS: ❍ 0: disabled, ❍ 1: enabled. • Parity: ❍ 00: none, ❍ 10: odd, ❍ 11: even. • Stop bit: ❍ 0: 1 stop bit, ❍ 1: 2 stop bits. • Data bits: ❍ 0: 7 data bits, ❍ 1: 8 data bits.	S, U

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW105 %SW106	Configuration for use of the ASCII protocol	When bit %S103 (SL1) or %S104 (SL2) is set to 1, the ASCII protocol is used. System word %SW105 (SL1) or %SW106 (SL2) must be set according to the elements below:  	S, U
%SW107 %SW108 %SW109	MAC address	Indicates the controller MAC address (only references with Ethernet channel). For MAC address AA:BB:CC:DD:EE:FF: ● %SW107 = AA:BB ● %SW108 = CC:DD ● %SW109 = EE:FF	S
%SW114	Enable schedule blocks	Enables or disables operation of schedule blocks by the program: ● Bit [0]: Enable/disable schedule block number 0 ❍ Set to 0: disabled ❍ Set to 1: enabled ● ... ● Bit [15]: Enable/disable schedule block number 15 ❍ Set to 0: disabled ❍ Set to 1: enabled Initially all schedule blocks are enabled. The default value is FFFF hex.	S and U, SIM
%SW115 %SW116 %SW117	Controller serial numbers part 1, 2, and 3 respectively (in BCD)	Allows to obtain the serial number of the controller. Example with the serial number 8A160400008: ● %SW115 : 16#0008 ● %SW116 : 16#6040 ● %SW117 : 16#0001	S

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW118	Logic controller status word	Indicates conditions on logic controller. For a controller operating normally, the value of this word is FFFF hex. ● Bit [9]: ❍ Set to 0: External error detected or communication interruption, for example duplicate IP address ❍ Set to 1: No error detected. ● Bit [10]: ❍ Set to 0: Invalid internal configuration; contact Schneider Electric customer service. ❍ Set to 1: No error detected. ● Bit [13]: ❍ 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. ❍ Set to 1: No error detected. ● Bit [14]: ❍ 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. ❍ Set to 1: No error detected. For more information on bus error handling, refer to I/O Configuration General Description *(see page 124)*. ● Bit [15]: ❍ Set to 0: Cartridge error detected (configuration or runtime operation). ❍ Set to 1: No error detected. **NOTE:** The other bits of this word are set to 1 and are reserved.	S, SIM
%SW119	**Optional module** feature configuration	One bit for each expansion module in the configuration: ● Bit [0]: Reserved for the logic controller ● Bit n: Module n ❍ Set to 1: Module is marked as optional in the configuration. ❍ Set to 0: Module is not marked as optional in the configuration.	S, SIM
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%SW120	Expansion I/O module status	1 bit for each expansion module in the configuration. Bit 0: Reserved for the logic controller When the logic controller attempts to start the I/O bus, bit n: ● 0 = no error detected ● 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). After the bus started and is running with data exchanges with the controller, bit n: ● 0 = no error detected ● 1 = error detected on the I/O expansion module (regardless if it is a module marked as optional). For more information on bus error handling, refer to I/O Configuration General Description *(see page 124)*.	S, SIM
%SW121 %SW122	Configuration for use of ASCII protocol	When bit %S103 (SL1) or %S104 (SL2) is set to 1, the ASCII protocol is used. You can change the ASCII frame size of SL1 or SL2. The ASCII frame size of SL1 is %SW121, and that of SL2 is %SW122.	U
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

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 µs of the event task associated with the input %I0.2.	S
%SW131	Event execution time	Indicates the last execution time in µs of the event task associated with the input %I0.3.	S
%SW132	Event execution time	Indicates the last execution time in µs of the event task associated with the input %I0.4.	S

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW133	Event execution time	Indicates the last execution time in µs of the event task associated with the input %I0.5.	S
%SW134	Event execution time	Indicates the last execution time in µs of the event task associated with the Threshold 0 of HSC0 or HSC2.	S
%SW135	Event execution time	Indicates the last execution time in µs of the event task associated with the Threshold 1 of HSC0 or HSC2.	S
%SW136	Event execution time	Indicates the last execution time in µs of the event task associated with the Threshold 0 of HSC1 or HSC3.	S
%SW137	Event execution time	Indicates the last execution time in µs 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 µs 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 **NOTE:** %SW139 is not used for sink outputs.	S
%SW140	Controller last error code 1	Most recent error code written to PlcLog.csv: AABBCCCCDD: %SW142 = AABB hex %SW141 = CCCC hex %SW140 = 00DD hex Where: • AA = error level • BB = error context • CCCC = error code • DD = error priority (internal use only)	S
%SW141	Controller last error code 2		
%SW142	Controller last error code 3		
%SW143	Number of entries in PlcLog.csv	Number of error codes contained in PlcLog.csv.	S

S Controlled by the system
U Controlled by the user
SIM Applied in the simulator

System Words	Function	Description	Control
%SW147	SD card operation result	If %SW90 set to 1, indicates that the SD card operation result after saving memory words. The error codes are: • 0: No error. • 1: Operation in progress. • 10: Eject the SD card. • 11: No SD card detected. • 12: SD card write protected • 13: The SD card is full. • 21: Number of memory words invalid. • 22: No memory words to be saved. • 30: A line in the CSV file is invalid. • 31: A line in the CSV file is too long. • 32: Format of the CSV file invalid. • 40: Error when creating the CSV file. • 50: Internal system error. • 51: Error when opening the CSV file.	S
%SW148	Number of persistent variables	• If %S90 is set to 0, you can save up to 2000 memory words (%MW50 up to %MW2049). • If %S90 is set to 1, you can save all memory words from %MW0. For more information, refer to Persistent Variables Saved by User Request *(see page 63)*.	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
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%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
%SW159	Event 0 average time	Indicates the average execution time in ms of the event task associated with the input %I0.2 (last 5 times).	S
%SW160	Event 1 average time	Indicates the average execution time in ms of the event task associated with the input %I0.3 (last 5 times).	S
%SW161	Event 2 average time	Indicates the average execution time in ms of the event task associated with the input %I0.4 (last 5 times).	S
%SW162	Event 3 average time	Indicates the average execution time in ms of the event task associated with the input %I0.5 (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	%SW167 indicates the status of the initialization command sent to the modem: ● If the modem does not respond to the initialization command within 10 attempts, its value is FFFF; modem does not respond. ● If the modem responds “OK” within the 10 attempts, its value is 0; modem is present and has accepted the initialization command. ● 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. **NOTE:** %S105 can be used to re-send the modem initialization command.	S
%SW168	Modbus TCP – Connections in use	Indicates the number of Ethernet Modbus TCP server connections in use. **NOTE:** 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.	S
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%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
%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: ● 0: Display not connected ● 1: Display application not ready ● 2: Display application transfer ● 3: Display application running ● 4: Display firmware update required ● 5: Display firmware transfer in progress	S
%SW183	Remote Graphic Display last error detected	Indicates the last error detected on the Remote Graphic Display: ● 0: No error detected ● 1: Display application transfer unsuccessful ● 2: Incompatible version of the display	S
S Controlled by the system **U** Controlled by the user **SIM** 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: • 112: **Setup Menu** • 113: **Controller Information** • 114: **Controller Setup** • 117: **Display Setup** • 120: **Controller State** • 121: **Controller Status** • 128: **Alarm View**	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
%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
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%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	Indicates the status of the EtherNet/IP class 1 connection: ● 0: At least one connection is idle. ● 1: The open connections are in run. ● 2: At least one connection has no indication or no communication. **NOTE:** Status 2 overrides status 0. **NOTE:** The application must be configured with a functional level *(see SoMachine Basic, Operating Guide)* of at least **Level 3.2** for this word to be supported.	S
%SW210	Status of the IOScanner SL1	Contains the status of the Modbus Serial IOScanner on Serial Line 1: ● 0: IOScanner is stopped ● 1: Initialization request to device being sent by IOScanner ● 2: IOScanner is operational ● 3: IOScanner is partially operational (some devices are not being scanned) ● 4: IOScanner is suspended	S
%SW211	Status of the IOScanner SL2	Contains the status of the Modbus Serial IOScanner on Serial Line 2: ● 0: IOScanner is stopped ● 1: Initialization request being sent by IOScanner ● 2: IOScanner is operational ● 3: IOScanner is partially operational (some devices are not being scanned) ● 4: IOScanner is suspended	S
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

System Words	Function	Description	Control
%SW212	Status of the Modbus TCP IOScanner	Contains the status of the Modbus TCP IOScanner on Ethernet: • 0: IOScanner is stopped • 1: Initialization request being sent by IOScanner to device • 2: IOScanner is operational • 3: IOScanner is partially operational (some devices are not being scanned) • 4: IOScanner is suspended **NOTE:** The application must be configured with a functional level *(see SoMachine Basic, Operating Guide)* of at least **Level 6.0** for this system word to be supported.	S
S Controlled by the system **U** Controlled by the user **SIM** Applied in the simulator			

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
TM221C16U	0x0796
TM221CE16U	0x0797
TM221C16T	0x0788
TM221CE16T	0x0789
TM221C24R	0x078A
TM221CE24R	0x078B
TM221C24T	0x078C
TM221CE24T	0x078D
TM221C24U	0x0798
TM221CE24U	0x0799
TM221C40R	0x078E

Reference	Code ID
TM221CE40R	0x078F
TM221C40T	0x0790
TM221CE40T	0x0791
TM221C40U	0x079A
TM221CE40U	0x079B

Input Channel Status (%IWS)

Introduction

The following provides information about the properties of input channel status words. A dedicated input channel status word exists for each analog input channel added using an I/O expansion module or TMC2 cartridge.

Displaying Input Channel Status Word Properties

Follow these steps to display the properties of the input channel status words:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **System objects → Input Status Words**. **Result**: Input channel status word properties is displayed.

Input Channel Status Word Properties

This table describes each property of the input channel status word:

Parameter	Editable	Value	Default Value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the input channel status word is being referenced in a program.
Address	No	%IWSx.y or %IWS0.x0y	–	The address of the input channel status word. For I/O expansion modules: ● x is the module number ● y is the channel number For analog cartridges: ● x is the cartridge number ● y is the channel number For example, `%IWS0.101` is the address of the second channel of the cartridge in the first slot of the logic controller.
Symbol	Yes	–	–	The symbol associated with the input channel status word. Double-click in the **Symbol** column and type the name of the symbol to associate with the input channel status word. If a symbol already exists, right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of the symbol throughout the program and/or program comments.

Parameter	Editable	Value	Default Value	Description
Comment	Yes	–	–	A comment associated with the input channel status word. Double-click in the **Comment** column and type an optional comment to associate with the input channel status word.

For More Information

To view the possible values of the input channel status word:

For information on:	refer to...
TM3 expansion modules	TM3 Analog I/O Modules Diagnostics *(see Modicon TM3 (SoMachine Basic), Expansion Modules Configuration, Programming Guide)*
TM2 expansion modules	TM2 Analog I/O Modules Diagnostics *(see Modicon TM2 (SoMachine Basic), Expansion Modules Configuration, Programming Guide)*
TMC2 cartridges	TMC2 Analog Cartridge Diagnostics *(see Modicon TMC2, Cartridges, Programming Guide)*

Output Channel Status (%QWS)

Introduction

The following provides information about the properties of output status words. A dedicated output channel status word exists for each analog output channel added using an I/O expansion module or TMC2 cartridge.

Displaying Output Channel Status Words Properties

Follow these steps to display the properties of the output channel status words:

Step	Action
1	Select the **Tools** tab in the left-hand area of the **Programming** window.
2	Click **System objects → Output Status Words**. **Result**: Output channel status word properties are displayed in the properties window.

Output Channel Status Word Properties

This table describes each property of the output channel status word:

Parameter	Editable	Value	Default Value	Description
Used	No	TRUE/FALSE	FALSE	Indicates whether the output channel status word is being referenced in a program.
Address	No	%QWSx.yor %QWS0.x0y	–	The address of the output channel status word. For I/O expansion modules: ● x is the module number ● y is the channel number For cartridges: ● x is the cartridge number ● y is the channel number For example, `%QWS3.0` is the address of the first output channel in the third I/O expansion module connected to the logic controller.
Symbol	Yes	–	–	The symbol associated with the output channel status word. Double-click in the **Symbol** column and type the name of the symbol to associate with the output channel status word. If a symbol already exists, right-click in the **Symbol** column and choose **Search and Replace** to find and replace occurrences of the symbol throughout the program and/or program comments.

Parameter	Editable	Value	Default Value	Description
Comment	Yes	–	–	A comment associated with the output channel status word. Double-click in the **Comment** column and type an optional comment to associate with the output channel status word.

For More Information

To view the possible values of the output channel status word:

For information on:	refer to...
TM3 expansion modules	TM3 Analog I/O Modules Diagnostics *(see Modicon TM3 (SoMachine Basic), Expansion Modules Configuration, Programming Guide)*
TM2 expansion modules	TM2 Analog I/O Modules Diagnostics *(see Modicon TM2 (SoMachine Basic), Expansion Modules Configuration, Programming Guide)*
TMC2 cartridges	TMC2 Analog Cartridge Diagnostics *(see Modicon TMC2, Cartridges, Programming Guide)*

Glossary

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.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

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.

F

FreqGen

(*frequency generator*) A function that generates a square wave signal with programmable frequency.

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.

HSC

(*high-speed counter*) A function that counts pulses on the controller or on expansion module inputs.

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.

PTO

(*pulse train outputs*) A fast output that oscillates between off and on in a fixed 50-50 duty cycle, producing a square wave form. PTO is especially well suited for applications such as stepper motors, frequency converters, and servo motor control, among others.

PWM

(*pulse width modulation*) A fast output that oscillates between off and on in an adjustable duty cycle, producing a rectangular wave form (though you can adjust it to produce a square wave).

R

RTC

(*real-time clock*) A battery-backed time-of-day and calender 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.

Index

Symbols

A

B

C

D

E

F

G

H

I

M

N

O

P

R

S

T

U

W