

Magelis SCU

HMI Controller

Programming Guide

12/2016



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

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

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

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

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

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

© 2016 Schneider Electric. All Rights Reserved.

Table of Contents



	Safety Information	5
	About the Book	7
Chapter 1	About the Magelis SCU HMI Controller	11
	Magelis SCU HMI Controller Devices Overview	11
Chapter 2	How to Configure the Controller	15
	How to Configure the HMI Controller	15
Chapter 3	Libraries	19
	Libraries	19
Chapter 4	Supported Standard Data Types	21
	Supported Standard Data Types	21
Chapter 5	Memory Mapping	23
	Memory Organization	23
Chapter 6	Tasks	27
	Maximum Number of Tasks	28
	Task Configuration	29
	Task Types	32
	Watchdogs	36
	Task Priorities	37
	Default Task Configuration	39
Chapter 7	Controller States and Behaviors	41
7.1	HMI Controller State Diagram	42
	Controller State Diagram	42
7.2	HMI Controller States Description	46
	Controller States Description	46
7.3	State Transitions and System Events	49
	Controller States and Output Behavior	50
	Commanding State Transitions	53
	Error Detection, Types, and Management	58
	Remanent Variables	59
Chapter 8	Controller Configuration	61
	Controller Parameters	62
	Controller Selection	63
	Applications	64
	PLC Settings	65

Chapter 9	Embedded Functions	67
	I/O Embedded Function	68
	HSC Embedded Function	72
	PTO_PWM Embedded Function	74
	Analog I/O Embedded Function	77
	Analog Temperature Embedded Function	82
Chapter 10	Ethernet Configuration	87
	IP Address Configuration	88
	Modbus TCP Client/Server	90
Chapter 11	Serial Line Configuration	91
	Serial Line Configuration	92
	Serial Line Protocol Manager	95
	SoMachine Network Manager	96
	Modbus Manager	97
Chapter 12	CANopen Configuration	99
	CANopen Interface Configuration	99
Chapter 13	Connecting the Magelis SCU HMI Controller to a PC ...	103
	Connecting the Controller to a PC	103
Chapter 14	Firmware Update	105
	Upgrading Magelis SCU HMI Controller Firmware/Runtime	105
Chapter 15	Magelis SCU HMI Controller - Troubleshooting and FAQ	107
	Troubleshooting	108
	Frequently Asked Questions	113
Appendices	123
Appendix A	Library SE_ModbusTCP_Slave	125
	Presentation of the Library SE_ModbusTCP_Slave	126
	Function block ModbusServer	127
Appendix B	Function and Function Block Representation	131
	Differences Between a Function and a Function Block	132
	How to Use a Function or a Function Block in IL Language	133
	How to Use a Function or a Function Block in ST Language	137
Appendix C	Controller Performance	141
	Processing Performance	141
Glossary	143
Index	151

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

The purpose of this document is to help you to configure and program your HMI SCU. For the programming of the HMI portion, refer to the Vijeo-Designer online help which can be found in SoMachine.

NOTE: Read and understand this document and all related documents before installing, operating, or maintaining your HMI SCU.

You should read through the entire document to understand all features.

Validity Note

This document has been updated for the release of SoMachine V4.2.

Related Documents

Title of Documentation	Reference Number
Magelis SCU HMI Controller PLCSystem Library Guide	EIO0000001246 (eng), EIO0000001247 (fre), EIO0000001248 (ger), EIO0000001249 (spa), EIO0000001250 (ita), EIO0000001251 (chs)
Magelis SCU HMI Controller HSC Library Guide	EIO0000001512 (eng), EIO0000001513 (fre), EIO0000001514 (ger), EIO0000001515 (spa), EIO0000001516 (ita), EIO0000001517 (chs)
Magelis SCU HMI Controller PTO/PWM Library Guide	EIO0000001518 (eng), EIO0000001519 (fre), EIO0000001520 (ger), EIO0000001521 (spa), EIO0000001522 (ita), EIO0000001523 (chs)

Title of Documentation	Reference Number
PLCCommunication Library Guide	EIO0000000361 (eng), EIO0000000742 (fre), EIO0000000743 (ger), EIO0000000744 (spa), EIO0000000745 (ita), EIO0000000746 (chs)
Magelis SCU HMI Controller Hardware Guide	EIO0000001232 (eng), EIO0000001233 (fre), EIO0000001234 (ger), EIO0000001235 (spa), EIO0000001236 (ita), EIO0000001237 (chs), EIO0000001238 (por)

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

Standard	Description
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.

Chapter 1

About the Magelis SCU HMI Controller

Magelis SCU HMI Controller Devices Overview

Overview

The Schneider Electric Magelis SCU HMI Controller has various powerful features. This controller can service a wide range of applications.

The software configuration and programming is accomplished with the SoMachine software.

Programming Languages

The Magelis SCU HMI Controller is supported and programmed with the SoMachine software, which supports the following IEC61131-3 programming languages:

- IL: Instruction List
- ST: Structured Text
- FBD: Function Block Diagram
- SFC: Sequential Function Chart
- LD: Ladder Diagram

SoMachine software can also be used to program these controllers using CFC (Continuous Function Chart) language.

Real Time Clock

The Magelis SCU HMI Controller includes a Real Time Clock (*see Magelis SCU, HMI Controller, Hardware Guide*) (RTC) system.

Run/Stop

The Magelis SCU HMI Controller can be operated externally by the following:

- a Run/Stop (*see Magelis SCU, HMI Controller, Hardware Guide*) operation by a dedicated digital input, defined in the software configuration. For more information, refer to Configuration of Digital Inputs (*see page 69*).
- a SoMachine software command.

Memory

The table shows the memory specifications:

Area	Element	Size (bytes)
System Area	System area reserved memory	131072
	System and diagnostic variables	
	Physical input addresses (%I)	256
	Physical output addresses (%Q)	256
	Retain variables ⁽¹⁾	16360
	Persistent retain variables	2044
Application Area	Compiled control application	1024000
User Area	Symbols	Dynamic allocation of 1024000
	Variables	
	Libraries	
(1) Not all of the 16360 bytes are available for the customer application because some libraries may use retain variables.		

Embedded Input/Output

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

- Regular inputs
- Fast inputs (HSC)
- Regular outputs
- Fast outputs (PTO/PWM)
- Analog inputs
- Analog outputs
- Thermocouple inputs
- RTD (Resistance Temperature Detector) inputs

Embedded Communication Features

4 types of communication ports are available on the rear panel:

- Ethernet port
- CANopen
- USB programming ports
- Serial link port

For more details, refer to the chapter Integrated Communication Ports (*see Magelis SCU, HMI Controller, Hardware Guide*).

Magelis SCU HMI Controller Range

Reference	Digital Input	Digital Output	Analog Input	Analog Output	Screen Size
HMISCU6A5	14 regular inputs and 2 fast inputs (HSC) ⁽¹⁾	8 regular outputs and 2 fast outputs (PTO) ⁽²⁾	No	No	8.9 cm (3.5 in.)
HMISCU8A5					14.48 cm (5.7 in.)
HMISAC					No
HMISCU6B5	6 regular inputs and 2 fast inputs (HSC) ⁽¹⁾	6 regular outputs and 2 fast outputs (PTO) ⁽²⁾	2 analog inputs (12-bit plus sign SAR ADC) and 2 analog inputs (16-bit), thermocouple, and RTD	2 analog outputs (12-bit)	8.9 cm (3.5 in.)
HMISCU8B5					14.48 cm (5.7 in.)
HMI SBC					No

(1) The fast inputs can be used either as regular inputs or as fast inputs for counting or event functions.
(2) The fast outputs can be used either as regular outputs or as fast outputs for PTO, PWM functions, or reflex output for HSC.

Chapter 2

How to Configure the Controller

How to Configure the HMI Controller

Introduction

First create a new project or open an existing project in the SoMachine software.

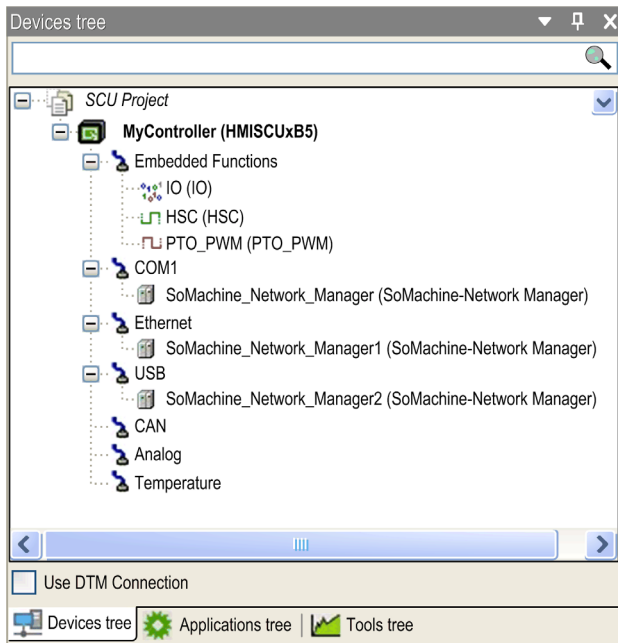
Refer to the SoMachine programming guide for information on how to:

- Add a controller to your project.
- Replace an existing controller.
- Convert a controller to a different but compatible device.

Devices Tree

The **Devices tree** shows a structured view of the current hardware configuration. When you add a controller to your project, a number of nodes are automatically added to the **Devices tree**, depending on the functions the controller provides.

Devices tree example:



The following is accessed from the **Devices tree** navigator:

Entry	Double-Click and Refer to...																																
HMISCUxx5	HMI Controller Device Editor (<i>see page 62</i>)																																
	<table border="1"> <thead> <tr> <th colspan="2" data-bbox="310 277 673 315">Embedded Functions</th> </tr> </thead> <tbody> <tr> <td data-bbox="310 315 488 352">IO</td> <td data-bbox="488 315 1254 352">I/O Embedded Function (<i>see page 68</i>) configuration</td> </tr> <tr> <td data-bbox="310 352 488 389">HSC</td> <td data-bbox="488 352 1254 389">HSC Embedded Function (<i>see page 72</i>) configuration</td> </tr> <tr> <td data-bbox="310 389 488 427">PTO_PWM</td> <td data-bbox="488 389 1254 427">PTO_PWM Embedded Function (<i>see page 74</i>) configuration</td> </tr> <tr> <td colspan="2" data-bbox="310 427 1254 464">COM1</td> </tr> <tr> <td colspan="2" data-bbox="310 464 1254 526">Serial Line (<i>see page 91</i>) configuration</td> </tr> <tr> <td colspan="2" data-bbox="310 526 1254 587">Ethernet⁽¹⁾</td> </tr> <tr> <td colspan="2" data-bbox="310 587 1254 647">An Ethernet connection is configured via Vijeo-Designer: Property Inspector → Download → Ethernet.</td> </tr> <tr> <td colspan="2" data-bbox="310 647 1254 709">USB⁽¹⁾</td> </tr> <tr> <td colspan="2" data-bbox="310 709 1254 771">A USB connection is configured via Vijeo-Designer: Property Inspector → Download → USB.</td> </tr> <tr> <td colspan="2" data-bbox="310 771 1254 797">CAN</td> </tr> <tr> <td colspan="2" data-bbox="310 797 1254 821">CANopen (<i>see page 99</i>) configuration</td> </tr> <tr> <td colspan="2" data-bbox="310 821 1254 847">Analog⁽²⁾</td> </tr> <tr> <td colspan="2" data-bbox="310 847 1254 873">Analog I/O Embedded Function (<i>see page 77</i>) configuration</td> </tr> <tr> <td colspan="2" data-bbox="310 873 1254 899">Temperature⁽²⁾</td> </tr> <tr> <td colspan="2" data-bbox="310 899 1254 925">Analog Temperature Embedded Function (<i>see page 82</i>) configuration</td> </tr> </tbody> </table>	Embedded Functions		IO	I/O Embedded Function (<i>see page 68</i>) configuration	HSC	HSC Embedded Function (<i>see page 72</i>) configuration	PTO_PWM	PTO_PWM Embedded Function (<i>see page 74</i>) configuration	COM1		Serial Line (<i>see page 91</i>) configuration		Ethernet⁽¹⁾		An Ethernet connection is configured via Vijeo-Designer: Property Inspector → Download → Ethernet.		USB⁽¹⁾		A USB connection is configured via Vijeo-Designer: Property Inspector → Download → USB.		CAN		CANopen (<i>see page 99</i>) configuration		Analog⁽²⁾		Analog I/O Embedded Function (<i>see page 77</i>) configuration		Temperature⁽²⁾		Analog Temperature Embedded Function (<i>see page 82</i>) configuration	
Embedded Functions																																	
IO	I/O Embedded Function (<i>see page 68</i>) configuration																																
HSC	HSC Embedded Function (<i>see page 72</i>) configuration																																
PTO_PWM	PTO_PWM Embedded Function (<i>see page 74</i>) configuration																																
COM1																																	
Serial Line (<i>see page 91</i>) configuration																																	
Ethernet⁽¹⁾																																	
An Ethernet connection is configured via Vijeo-Designer: Property Inspector → Download → Ethernet.																																	
USB⁽¹⁾																																	
A USB connection is configured via Vijeo-Designer: Property Inspector → Download → USB.																																	
CAN																																	
CANopen (<i>see page 99</i>) configuration																																	
Analog⁽²⁾																																	
Analog I/O Embedded Function (<i>see page 77</i>) configuration																																	
Temperature⁽²⁾																																	
Analog Temperature Embedded Function (<i>see page 82</i>) configuration																																	
(1) For more information on how to configure the connection between your computer and the HMI controller, refer to Vijeo-Designer online help.																																	
(2) Only available on HMISCU6B5, HMISCU8B5 including HMI SBC.																																	

Applications Tree

The **Applications tree** allows you to manage project-specific applications as well as global applications, POUs, and tasks.

Tools Tree

The **Tools tree** allows you to configure the HMI part of your project and to manage libraries.

Chapter 3

Libraries

Libraries

Introduction

Libraries provide functions, function blocks, data types and global variables that can be used to develop your project.

The **Library Manager** of SoMachine provides information about the libraries included in your project and allows you to install new ones. For more information on the **Library Manager**, refer to the Functions and Libraries User Guide.

Magelis SCU HMI Controller

When you select a Magelis SCU HMI Controller for your application, SoMachine automatically loads the following libraries:

Library name	Description
IoStandard	CmploMgr configuration types, ConfigAccess , Parameters and help functions: manages the I/Os in the application.
Standard	Contains functions and function blocks which are required matching IEC61131-3 as standard POU's for an IEC programming system. Link the standard POU's to the project (standard.library).
Util	Analog Monitors, BCD Conversions, Bit/Byte Functions, Controller Datatypes, Function Manipulators, Mathematical Functions, Signals.
PLCCommunication (see <i>SoMachine, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide</i>)	SysMem, Standard, SE_PLCSysMem . These functions facilitate communications between specific devices. Most of them are dedicated to Modbus exchange. Communication functions are processed asynchronously regarding the application task that called the function.
HMISCU PLCSystem (see <i>Magelis SCU, HMI Controller, PLCSystem Library Guide</i>)	Contains functions and variables to get information and send commands to the controller system.

Library name	Description
HMISCU HSC (<i>see Magelis SCU, HMI Controller, HSC Library Guide</i>)	Contains function blocks and variables to get information and send commands to the Fast Inputs/Outputs of the Magelis SCU HMI Controller. These function blocks permit you to implement HSC (High Speed Counting) functions on the Fast Inputs/Outputs of the Magelis SCU HMI Controller.
HMISCU PTO/PWM (<i>see Magelis SCU, HMI Controller, PTO/PWM Library Guide</i>)	Contains function blocks and variables to get information and send commands to the Fast Inputs/Outputs of the Magelis SCU HMI Controller. These function blocks permit you to implement PTO (Pulse Train Output) and PWM (Pulse Width Modulation) functions on the Fast Outputs of the Magelis SCU HMI Controller.

Those additional libraries can also be added for your Magelis SCU HMI Controller application:

Library name	Description
<ul style="list-style-type: none"> ● <code>_3SCOS</code>: 3S CANopenStack ● <code>FDT_CAN</code>: <code>FDT_CANOpenDriver</code> ● <code>CIA405</code>: CAA CiA 405 	<p>The CAA CiA 405 Library offers a set of function blocks to meet the requirements of the CiA405 for the access to the CANopen network from the application (IEC61131-3 program) of the controller (CANopen master).</p> <p>NOTE: These libraries are not loaded automatically when you add the HMI SCU device. They are added when the CANopen_Optimized child node under CAN is added.</p>
<code>SE_ModbusTCP_Slave</code>	Contains a function block for managing the communication between the HMI SCU controller in the capacity of a Modbus server and any clients requesting Modbus services from the controller. For more details about this library, refer to Library <code>SE_ModbusTCP_Slave</code> (<i>see page 125</i>).

Chapter 4

Supported Standard Data Types

Supported Standard Data Types

Supported Standard Data Types

The controller supports the following IEC data types:

Data Type	Lower Limit	Upper Limit	Information Content
BOOL	FALSE	TRUE	1 Bit
BYTE	0	255	8 Bit
WORD	0	65,535	16 Bit
DWORD	0	4,294,967,295	32 Bit
LWORD	0	$2^{64}-1$	64 Bit
SINT	-128	127	8 Bit
USINT	0	255	8 Bit
INT	-32,768	32,767	16 Bit
UINT	0	65,535	16 Bit
DINT	-2,147,483,648	2,147,483,647	32 Bit
UDINT	0	4,294,967,295	32 Bit
LINT	-2^{63}	$2^{63}-1$	64 Bit
ULINT	0	$2^{64}-1$	64 Bit
REAL	1.175494351e-38	3.402823466e+38	32 Bit
LREAL	2.2250738585072014e-308	1.7976931348623158e+308	64 Bit
STRING	1 character	255 characters	1 character = 1 byte
WSTRING	1 character	255 characters	1 character = 1 word
TIME	-	-	32 Bit

For more information on ARRAY, LTIME, DATE, TIME, DATE_AND_TIME, and TIME_OF_DAY, refer to the SoMachine Programming Guide.

Chapter 5

Memory Mapping

Memory Organization

Introduction

This section describes the memory size for different areas of the Magelis SCU HMI Controller.

Memory Mapping

The memory size of the controllers is 128 Mbytes.

The table presents the memory specifications of the non-HMI control component:

Area	Element	Size (bytes)
System Area	System area reserved memory	131072
	System and diagnostic variables	
	Physical input addresses (%I)	256
	Physical output addresses (%Q)	256
	Retain variables ⁽¹⁾	16360
	Persistent retain variables	2044
Application Area	Compiled control application	1024000
User Area	Symbols	Dynamic allocation of 1228800
	Variables	
	Libraries	
(1) Not all of the 16360 bytes are available for the customer application because some libraries may use retain variables.		

The memory is composed of four areas:

- controller dedicated application memory
- controller OS memory
- HMI application memory
- HMI OS memory

Memory containing Persistent and Retain variables is preserved and protected. The Persistent and Retain variables will be retained during power outages or when the HMI controller is powered off.

System and Diagnostic Variables

Variables	Description
PLC_R	Structure of HMI controller read-only system variables.
PLC_W	Structure of HMI controller read/write system variables.
ETH_R	Structure of Ethernet read-only system variables.
ETH_W	Structure of Ethernet read/write system variables.
SERIAL_R	Structure of Serial Lines read-only system variables.
SERIAL_W	Structure of Serial Lines read/write system variables.

For more information on System Variables, refer to the *Magelis SCU SoMachine PLCSystem Library Guide* (see *Magelis SCU, HMI Controller, PLCSystem Library Guide*).

Library Sizes

Library Name	Average Size	Comment
HMISCU HSC (see <i>Magelis SCU, HMI Controller, HSC Library Guide</i>)	10 kbytes	Depends on the functions used.
HMISCU PLCSystem (see <i>Magelis SCU, HMI Controller, PLCSystem Library Guide</i>)	25 kbytes	Always embedded in the application. The use of the functions does not consume additional memory.
HMISCU PTO/PWM (see <i>Magelis SCU, HMI Controller, PTO/PWM Library Guide</i>)	10 kbytes	Depends on the functions used.
PLC Communication	20 kbytes	Depends on the functions used.
CANopen Stack	115 kbytes	Depends on the functions used. Each CANopen Slave consumes approximately an additional 10 kbytes of memory.
SE_ModbusTCP_Slave (see page 125)	23 kbytes	Depends on the number of connections.

Differences Between Byte Addressing and Word Oriented IEC Addressing

See the table below for a comparison of byte addressing and word-oriented IEC addressing for bits, bytes, words, and dwords. It visualizes the overlapping memory areas in case of byte addressing mode (see the example below the table).

Concerning the notation, consider that, for bit addresses, the IEC addressing mode is always word-oriented. This means that the place before the dot corresponds to the number of the word, the place behind names the number of the bit.

Comparison of byte and word oriented addressing for the address sizes D , W , B and X :

DWORDS/WORDS		Bytes	X (bits)		
D0	W0	B0	x0.7	...	x0.0
		B1	x1.7	...	x1.0
	W1	B2			
		B3			
D1	W2	B4			
		B5			
	W3	B6			
		B7			
D2	W4	B8			
		...			
			
		...			

Example for overlapping of memory ranges in case of byte addressing mode:

- D0 contains B0 . . . B3
- W0 contains B0 and B1
- W1 contains B2 and B3
- W2 contains B4 and B5

In order to get around the overlap do not use W1 or D1, D2, D3 for addressing.

Chapter 6

Tasks

Introduction

The **Task Configuration** node in the **Applications tree** allows you to define one or more tasks to control the execution of your application program.

The task types available are:

- Cyclic
- Freewheeling
- Event
- External event

This chapter begins with an explanation of these task types and provides information regarding the maximum number of tasks, the default task configuration, and task prioritization. In addition, this chapter introduces the task watchdog function and explains its relationship to task execution.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Maximum Number of Tasks	28
Task Configuration	29
Task Types	32
Watchdogs	36
Task Priorities	37
Default Task Configuration	39

Maximum Number of Tasks

Maximum Number of Tasks

The maximum number of tasks you can define for the Magelis SCU HMI Controller are:

- Total number of tasks = 7
- Cyclic tasks = 3
- Freewheeling tasks = 1
- Event tasks = 2
- External tasks = 2

Special Considerations for Freewheeling

A Freewheeling task (*see page 33*) does not have a fixed duration. In Freewheeling mode, each task scan starts when the previous scan has been completed and after a period of system processing (30% of the total duration of the Freewheeling task).


NOTE: You may wish to avoid using a Freewheeling task in a multi-task application when some high priority and time-consuming tasks are running. Doing so may provoke a task Watchdog Timeout. You should not assign CANopen to a freewheeling task. CANopen should be assigned to a cyclic task.

NOTE: This does not include the CPU process time given to process the HMI application.

Task Configuration

Adding Tasks

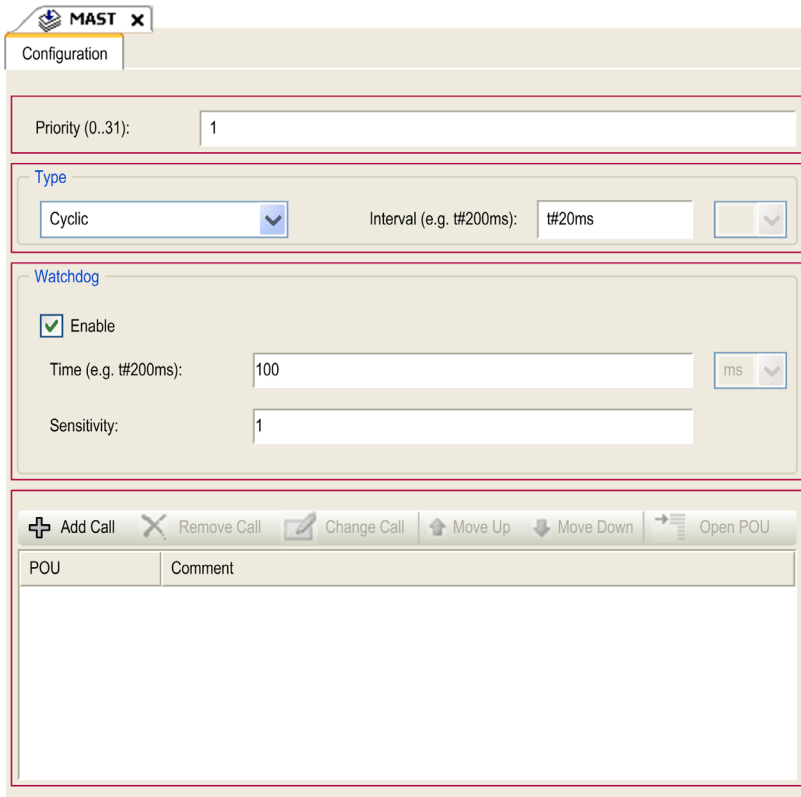
You can add tasks to your application via the **Applications tree**:

Step	Action
1	In the Applications tree , select the Task Configuration node.
2	Click  .
3	Select Task.... Result: The Add Task window box opens.
4	In the Add Task window box, enter a name in the Name: text box. Note: The name must neither contain any space nor exceed a length of 32 characters.
5	Click Add .

Task Configuration Dialog

Each task configuration has its own parameters which are independent of the other tasks.

The task **Configuration** dialog is composed of 4 parts:



The table describes the fields of the **Task Configuration** window:

Field Name	Definition
Priority	<p>You can configure the priority of each task with a number from 0 to 31 (0 is the highest priority, 31 is the lowest).</p> <p>Only one task at a time can be running. The priority determines when the task runs:</p> <ul style="list-style-type: none"> • a higher priority task preempts a lower priority task • tasks with same priority run in turn (2 ms time-slice) <p>NOTE: Do not assign tasks with the same priority. If there are yet other tasks that attempt to preempt tasks with the same priority, the result could be indeterminate and unpredictable. For more important safety information, refer to Task Priorities (see page 37).</p>

Field Name	Definition
Type	<p>These task types are available:</p> <ul style="list-style-type: none"> ● Cyclic (<i>see page 32</i>) ● Event (<i>see page 34</i>) ● External (<i>see page 34</i>) ● Freewheeling (<i>see page 33</i>)
Watchdog	<p>To configure the watchdog (<i>see page 36</i>), define the following 2 parameters:</p> <ul style="list-style-type: none"> ● Time: enter the timeout before watchdog execution. ● Sensitivity: defines the number of expirations of the watchdog timer before the controller stops program execution and enters a HALT state.
POUs	<p>The list of POUs (Programming Organization Units) controlled by the task is defined in the task configuration window:</p> <ul style="list-style-type: none"> ● To add a POU linked to the task, use the command Add Call and select the POU in the Input Assistant editor. ● To remove a POU from the list, use the command Remove Call. ● To replace the currently selected POU of the list by another one, use the command Change Call. ● POUs are executed in the order shown in the list. To move the POUs in the list, select a POU and use the command Move Up or Move Down. <p>NOTE: You can create as many POUs as you want. An application with several small POUs, as opposed to one large POU, can improve the refresh time of the variables in online mode.</p>

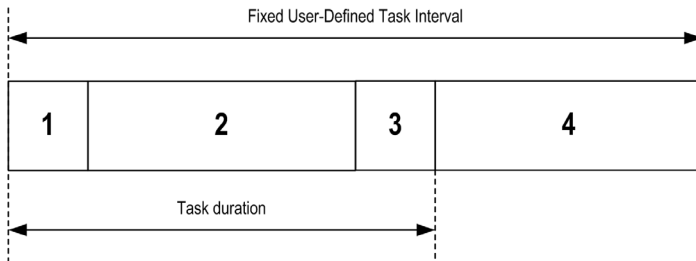
Task Types

Introduction

The following section describes the various task types available for your program, along with a description of the task type characteristics.

Cyclic Task

A Cyclic task is assigned a fixed cycle time using the Interval setting in the Type section of Configuration subtab for that task. Each Cyclic task type executes as follows:



1. **Read Inputs:** The physical input states are written to the %I input memory variables and other system operations are executed.
2. **Task Processing:** The user code (POU, and so on) defined in the task is processed. The %Q output memory variables are updated according to your application program instructions but not yet written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variables are modified with any output forcing that has been defined; however, the writing of the physical outputs depends upon the type of output and instructions used.
For more information on defining the bus cycle task, refer to the SoMachine Programming Guide and Magelis SCU HMI Controller Settings.
For more information on I/O behavior, refer to Controller States Detailed Description (*see page 46*).
4. **Remaining Interval time:** The controller firmware carries out system processing and any other lower priority tasks.

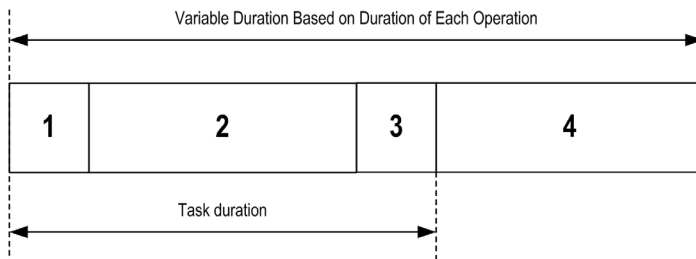
NOTE: If you define too short a period for a cyclic task, it will repeat immediately after the write of the outputs and without executing other lower priority tasks or any system processing. This will affect the execution of all tasks and cause the controller to exceed the system watchdog limits, generating a system watchdog exception.

NOTE: The task cycle time is set to a value greater than or equal to 4 ms and the task interval is a multiple of 4 ms.

NOTE: Get and set the interval of a Cyclic Task by application using the **GetCurrentTaskCycle** and **SetCurrentTaskCycle** function. (Refer to Toolbox Advance Library Guide for further details.)

Freewheeling Task

A Freewheeling task does not have a fixed duration. In Freewheeling mode, each task scan begins when the previous scan has been completed and after a short period of system processing. Each Freewheeling task type executes as follows:




1. **Read Inputs:** The physical input states are written to the %I input memory variables and other system operations are executed.
2. **Task Processing:** The user code (POU, and so on) defined in the task is processed. The %Q output memory variables are updated according to your application program instructions but not yet written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variables are modified with any output forcing that has been defined; however, the writing of the physical outputs depends upon the type of output and instructions used.
For more information on defining the bus cycle task, refer to the SoMachine Programming Guide and Magelis SCU HMI Controller Settings.
For more information on I/O behavior, refer to Controller States Detailed Description (*see page 46*).
4. **System Processing:** The controller firmware carries out system processing and any other lower priority tasks (for example: HTTP management, Ethernet management, parameters management).

Event Task

This type of task is event-driven and is initiated by a program variable. It starts at the rising edge of the boolean variable associated to the trigger event unless pre-empted by a higher priority task. In that case, the Event task will start as dictated by the task priority assignments.

For example, if you have defined a variable called `my_Var` and would like to assign it to an Event, proceed as follows:

Step	Action
1	Double-click the TASK in the Applications tree .
2	Select Event from the Type list in the Configuration tab.
3	Click the Input Assistant button  to the right of the Event field. Result: The Input Assistant window appears.
4	Navigate in the tree of the Input Assistant dialog box to find and assign the <code>my_Var</code> variable.

NOTE: The maximum frequency admissible for the event triggering an Event task is governed by the priorities of other tasks and system processes. So you must test your application to ensure reliable event triggering.

External Event Task

This type of task is event-driven and is initiated by the detection of a hardware or hardware-related function event. It starts when the event occurs unless pre-empted by a higher priority task. In that case, the External Event task will start as dictated by the task priority assignments.

For example, an `External Event Task` can be associated with an HSC Threshold cross event. To associate the `HSC0_TH1` event to an External Event task, select it from the **External event** drop-down list on the **Configuration** sub-tab.

For the Magelis HMI SCU controller, there are 2 types of events that can be associated with an `External Event Task`:

- a FAST input (FI0 and FI1) on a rising edge, falling edge, or both edges
- an HSC threshold when counting up, counting down, or both counting up and down

External Event Task: Performance

For an `External Event Task` triggered by `F10`, `F11`, `HSC0_TH0`, or `HSC0_TH1`, the minimum stable interval between triggers is:

- 1.5 ms for tasks that do not require an immediate state change to FAST outputs (`FQ0` or `FQ1`)
- 15 ms for tasks that require an immediate state change to FAST outputs (`FQ0` or `FQ1`)

If the triggering conditions are met, but at a shorter interval than listed above, `External Event Task` execution will be subject to delays, or may not execute. Complex tasks that require more computational time than the above times may also lead to `External Event Tasks` being subject to delays or missed execution.

WARNING

UNINTENDED EQUIPMENT OPERATION

Test your application thoroughly to ensure that your application performance meets your specifications.

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

Watchdogs

Introduction

Two types of watchdog functionality are implemented for the Magelis SCU HMI Controller:

- **Task Watchdogs:** These watchdogs are optional watchdogs that you can define for each task. These are managed by your application program and are configurable in SoMachine.
- **Hardware Watchdogs:** This watchdog is managed by the HMI controller main CPU. It is not configurable by the user.

NOTE: Infinite loops are not interrupted if a user does not set the Task Watchdog manually.

Task Watchdogs

SoMachine allows you to configure an optional task watchdog for every task defined in your application program. (Task watchdogs are sometimes also referred to as software watchdogs or control timers in the SoMachine online help). When one of your defined task watchdogs reaches its threshold condition, an application error is detected and the controller enters the HALT state.

When defining a task watchdog, the following options are available:

- **Time:** This defines the allowable maximum execution time for a task. When a task takes longer than this, the controller will report a task watchdog exception.
- **Sensitivity:** The sensitivity field defines the number of task watchdog exceptions that must occur before the controller detects an application error.

To access the configuration of a task watchdog, double-click the **Task** in the **Applications tree**.

NOTE: For more information on watchdogs, refer to SoMachine Programming Guide.

Hardware Watchdog

If the main CPU of an HMISCU processes invalid instructions (application corruption or invalid commands) or does not respond for 10 seconds, the Hardware Watchdog is triggered. This results in a software reset of the Vijeo Designer Runtime. After the reset, no controller application is loaded and the Vijeo Designer is in a **Ready for Download** state.

A new HMI and Controller application must be downloaded to the device to recover it from this condition.

Task Priorities

Task Priority Configuration

You can configure the priority of each task between 0 and 31 (0 is the highest priority, 31 is the lowest). Each task must have a unique priority. If you assign the same priority to more than one task, execution for those tasks is indeterminate and unpredictable, which may lead to unintended consequences.

WARNING

UNINTENDED EQUIPMENT OPERATION

Do not assign the same priority to different tasks.

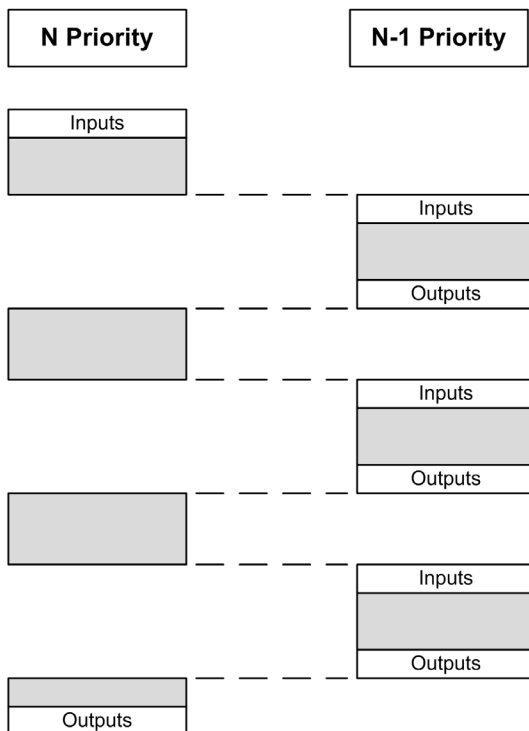
Failure to follow these instructions can result in death, serious injury, or equipment damage.

Task Priority Suggestions

- Priority 0 to 24: Controller tasks. Assign these priorities to tasks with a high availability requirement.
- Priority 25 to 31: Background tasks. Assign these priorities to tasks with a low availability requirement.

Task Preemption Due to Task Priorities

When a task cycle starts, it can interrupt any task with lower priority (task preemption). The interrupted task will resume when the higher priority task cycle is finished.



NOTE: If the same input is used in different tasks the input image may change during the task cycle of the lower priority task.

To improve the likelihood of proper output behavior during multitasking, a message is displayed if outputs in the same byte are used in different tasks.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Map your inputs so that tasks do not alter the input images in an unexpected manner.

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

Default Task Configuration

Default Task Configuration

The MAST task can be configured in Freewheeling or Cyclic mode. The MAST task is automatically created by default in Cyclic mode. Its preset priority is medium (15), its preset interval is 20 ms, and its task watchdog service is activated with a time of 100 ms and a sensitivity of 1. Refer to Task Priorities (*see page 37*) for more information on priority settings. Refer to Task Watchdogs (*see page 36*) for more information on watchdogs.

Designing an efficient application program is important in systems approaching the maximum number of tasks. In such an application, it can be difficult to keep the resource utilization below the system watchdog threshold. If priority reassignments alone are not sufficient to remain below the threshold, some lower priority tasks can be made to use fewer system resources if the SysTaskWaitSleep function is added to those tasks. For more information about this function, see the optional SysTask library of the system / SysLibs category of libraries.

NOTE: Do not delete or change the name of the MAST task. Otherwise, SoMachine detects an error when you attempt to build the application, and you will not be able to download it to the controller.

Chapter 7

Controller States and Behaviors

Introduction

This chapter 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 Remanent variables and the effect of SoMachine task programming options on the behavior of your system.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
7.1	HMI Controller State Diagram	42
7.2	HMI Controller States Description	46
7.3	State Transitions and System Events	49

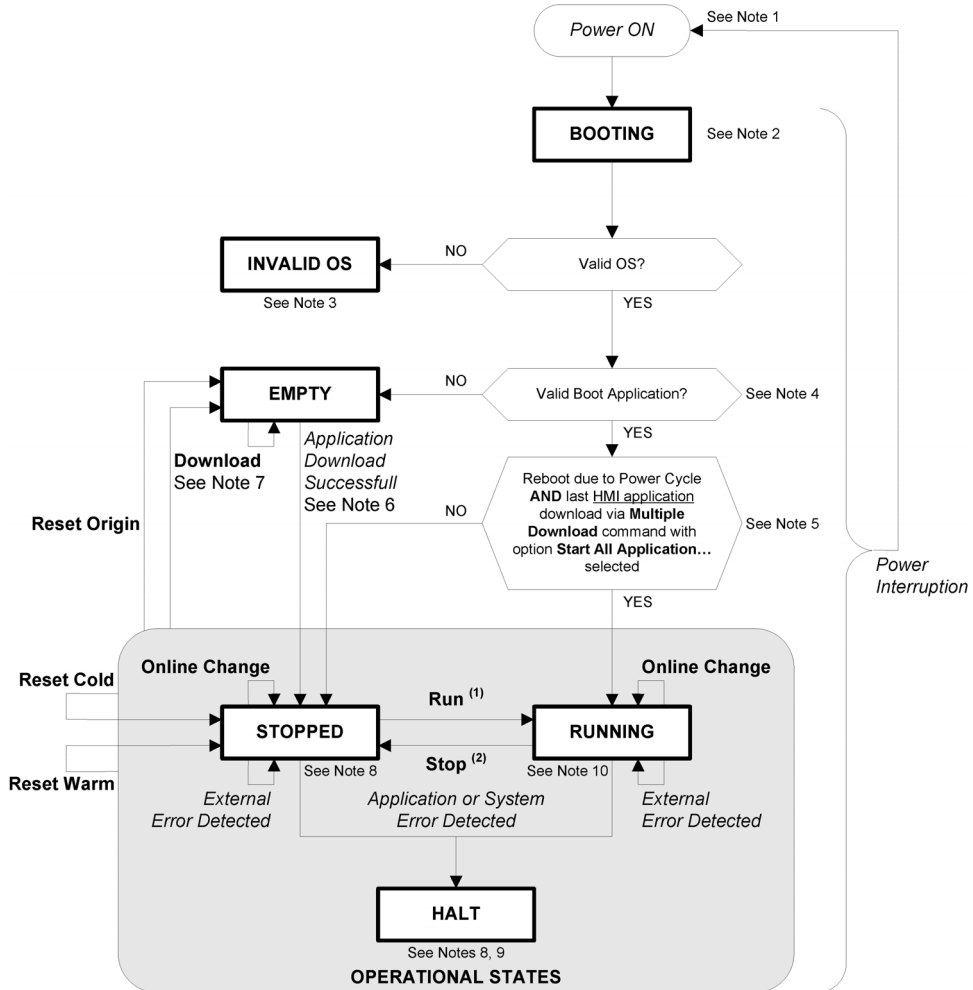
Section 7.1

HMI Controller State Diagram

Controller State Diagram

Controller State Diagram

The following diagram describes the controller operating mode:



Legend:

- Controller states are indicated in **ALL-CAPS BOLD**
- User and application commands are indicated in **Bold**
- System events are indicated in *Italics*
- Decisions, decision results and general information are indicated in normal text

(1) For details on STOPPED to RUNNING state transition, refer to Run Command (*see page 53*).

(2) For details on RUNNING to STOPPED state transition, refer to Stop Command (*see page 53*).

Note 1

The Power Cycle (Power Interruption followed by a Power ON) deletes all output forcing settings. Refer to Controller State and Output Behavior for further details.

Note 2

The outputs will assume their initialization states.

Note 3

HMI download screen is displayed prompting the user to download the firmware, HMI and Control application.

Note 4

The application is loaded into RAM after verification of a valid Boot application.

Note 5

The state of the controller will be RUNNING after a reboot if the reboot was provoked by a Power Cycle and the **HMI application** had been downloaded using a **Multiple Download...** command with option **Start all applications after download or online change** selected.

Note 6

During a successful application download the following events occur:

- The application is loaded directly into RAM.
- By default, the Boot application is created and saved into the Flash memory.

Note 7

However, there are two important considerations in this regard:

- **Online Change:** An online change (partial download) initiated while the controller is in the RUNNING state returns the controller to the RUNNING state if successful.
Before using the **Login with online change** option, test the changes to your application program in a virtual or non-production environment and confirm that the controller and attached equipment assume their expected conditions in the RUNNING state.

WARNING

UNINTENDED EQUIPMENT OPERATION

Always verify that online changes to a RUNNING application program operate as expected before downloading them to controllers.

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

NOTE: Online changes to your program are not automatically written to the Boot application, and will be overwritten by the existing Boot application at the next reboot. If you wish your changes to persist through a reboot, manually update the Boot application by selecting **Create boot application** in the Online menu.

- **Multiple Download:** SoMachine has a feature that allows you to perform a full application download to multiple targets on your network or fieldbus.
One of the default options when you select the **Multiple Download...** command is the **Start all applications after download or online change** option, which restarts all download targets in the RUNNING state, irrespective of their last controller state before the multiple download was initiated. Deselect this option if you do not want all targeted controllers to restart in the RUNNING state.
In addition, before using the **Multiple Download...** option, test the changes to your application program in a virtual or non-production environment and confirm that the targeted controllers and attached equipment assume their expected conditions in the RUNNING state.

WARNING

UNINTENDED EQUIPMENT OPERATION

Always verify that your application program will operate as expected for all targeted controllers and equipment before issuing the **Multiple Download...** command with the **Start all applications after download or online change** option selected.

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

Note 8

The SoMachine software platform allows many powerful options for managing task execution and output conditions while the controller is in the STOPPED or HALT states. Refer to Controller State and Output Behavior for further details.

Note 9

To exit the HALT state it is necessary to issue one of the Reset commands (Reset Warm, Reset Cold, Reset Origin), download an application or cycle power.

In the event a Hardware Watchdog is triggered, an automatic reboot into **Ready for Download** mode occurs. In this state, the HMI application and the controller application are not loaded. The device can be recovered by downloading new HMI and controller applications.

Note 10

The RUNNING state has two exceptional conditions that will be indicated in run state or error messages on HMI screen.

- RUNNING with External Error: You may exit this exceptional condition by clearing the external error. No controller commands are required.
- RUNNING with Breakpoint: Refer to Controller State Description (*see page 46*) for further details on this exceptional condition.


Section 7.2

HMI Controller States Description

Controller States Description

Introduction

This section provides a detailed description of the controller states.

 WARNING
UNINTENDED EQUIPMENT OPERATION
<ul style="list-style-type: none"> • 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 verifying the presence of output forcing, and reviewing the controller status information via SoMachine ⁽¹⁾.
Failure to follow these instructions can result in death, serious injury, or equipment damage.

⁽¹⁾ **Note:** The controller states can be read in the PLC_R.i_wStatus system variable of the PLCSystem library (see *Magelis SCU, HMI Controller, PLCSystem Library Guide*).

Controller States Table

This table describes the controller states:

Controller State	Description
BOOTING	The controller executes the boot firmware and its own internal self-tests. It then checks the checksum of the firmware and user applications. It does not execute the application nor does it communicate.
INVALID_OS	There is not a valid firmware file present in the Flash memory. The controller does not execute the application. Communication is only possible through the USB host port, and then only for uploading a valid OS.
EMPTY	There is no application in memory or the application is invalid.
RUNNING	The controller is executing a valid application.
RUNNING with Breakpoint	This state is the same as the RUNNING state with the following exceptions: <ul style="list-style-type: none"> • The task-processing portion of the program does not resume until the breakpoint is cleared. For more information, refer to breakpoints management.

Controller State	Description
RUNNING with detection of an <i>External Error</i>	This state is the same as the normal RUNNING state.
STOPPED	The controller has a valid application that is stopped. See Details of the STOPPED State (see page 47) for an explanation of the behavior of outputs and field buses in this state.
STOPPED with detection of an <i>External Error</i>	This state is the same as the normal STOPPED state.
HALT	The controller stops executing the application because it has detected an Application or a System Error. This description is the same as for the STOPPED state with the following exceptions: <ul style="list-style-type: none"> • The task responsible for the Application Error always behaves as if the Update I/O while in stop option was not selected. All other tasks follow the actual setting.

Details of the STOPPED State

The following statements are always true for the STOPPED state:

- The input configured as the Run/Stop input remains operational.
- Ethernet, Serial (Modbus, ASCII, and so on), and USB communication services remain operational and commands written by these services can continue to affect the application, the controller state, and the memory variables.
- All outputs initially assume their configured state (**Keep current values** or **Set all outputs to default**) or the state dictated by output forcing if used. The subsequent state of the outputs depends on the value of the **Update I/O while in stop** setting and on commands received from remote devices.

Task and I/O Behavior When Update I/O While In Stop Is Selected

When the **Update I/O while in stop** setting is selected:

- The Read Inputs operation continues normally. The physical inputs are read and then written to the %I input memory variable.
- The Task Processing operation is not executed.
- The Write Outputs operation continues. The %Q output memory variable is updated to reflect either the **Keep current values** configuration or the **Set all outputs to default** configuration, adjusted for any output forcing, and then written to the physical outputs.

NOTE: Expert functions continue to operate. For example, a counter will continue to count. However, these Expert functions do not affect the state of the outputs. The outputs of Expert I/O conform to the behavior stated here.

NOTE: Commands received by Ethernet, Serial, USB, and CAN communications can continue to write to the memory variables. Changes to the %Q output memory variables are written to the physical outputs.

CAN Behavior When Update I/O While In Stop Is Selected

The following is true for the CAN buses when the Update I/O while in stop setting is selected:

- The CAN bus remains fully operational. Devices on the CAN bus continue to perceive the presence of a functional CAN Master.
- TPDO and RPDO continue to be exchanged.
- The optional SDO, if configured, continue to be exchanged.
- The Heartbeat and Node Guarding functions, if configured, continue to operate.
- If the **Behavior for outputs in Stop** field is set to **Keep current values**, the TPDOs continue to be issued with the last actual values.
- If the **Behavior for outputs in Stop** field is **Set all outputs to default** the last actual values are updated to the default values and subsequent TPDOs are issued with these default values.

Task and I/O Behavior When Update I/O While In Stop Is Not Selected

When the **Update I/O while in stop** setting is not selected, the controller sets the I/O to either the **Keep current values** or **Set all outputs to default** condition (as adjusted for output forcing if used). After this, the following becomes true:

- The Read Inputs operation ceases. The %I input memory variable is frozen at its last values.
- The Task Processing operation is not executed.
- The Write Outputs operation ceases. The %Q output memory variables can be updated via the Ethernet, Serial, and USB connections. However, the physical outputs are unaffected and retain the state specified by the configuration options.

NOTE: Expert functions cease operating. For example, a counter will be stopped.

CAN Behavior When Update I/O While In Stop Is Not Selected

The following is true for the CAN buses when the **Update I/O while in stop** setting is not selected:

- The CAN Master ceases communications. Devices on the CAN bus assume their configured fallback states.
- TPDO and RPDO exchanges cease.
- Optional SDO, if configured, exchanges cease.
- The Heartbeat and Node Guarding functions, if configured, stop.
- The current or default values, as appropriate, are written to the TPDOs and sent once before stopping the CAN Master.

Section 7.3

State Transitions and System Events

Overview

This section begins with an explanation of the output states possible for the controller. It then presents the system commands used to transition between controller states and the system events that can also affect these states. It concludes with an explanation of the Remanent variables, and the circumstances under which different variables and data types are retained through state transitions.

What Is in This Section?

This section contains the following topics:

Topic	Page
Controller States and Output Behavior	50
Commanding State Transitions	53
Error Detection, Types, and Management	58
Remanent Variables	59

Controller States and Output Behavior

Introduction

The Magelis SCU HMI 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. For example, typical controllers define only 2 options for output behavior in stop: fallback to default value or keep current value.

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

- managed by **Application Program**
- keep **Current Values**
- set All **Outputs to Default**
- hardware **Initialization Values**
- software **Initialization Values**
- **Output Forcing**

ControllerLockout Feature

The **ControllerLockout** feature locks or unlocks the controller stop mode. A locked controller cannot be restarted until the controller is unlocked.

Attempts to restart a locked controller are ignored and a message appears. You can only initiate lockout once the controller is in STOPPED state. If the controller is in RUNNING state and you attempt to lockout, the attempt is ignored and a message appears.

The **ControllerLockout** is not managed through SoMachine, it is an internal boolean variable (ControllerLockout) of the HMI in Vijeo-Designer.

For more information on managing this variable, refer to the Vijeo-Designer Online Help.

Managed by Application Program

Your application program manages outputs normally. This applies in the RUNNING and RUNNING with External Error Detected states.

Keep Current Values

Select this option by choosing **Keep current values** in the **Behavior for outputs in Stop** drop-down menu of the **PLC settings** subtab of the **Controller Editor**. To access the Controller Editor, right-click on the controller in the device tree and select **Edit Object**.

Double-click the name representing the HMISCU device in the **Devices** windows to access the **Controller Editor**.

This output behavior applies in the STOPPED controller state. It also applies to CAN bus in the HALT controller state. Outputs are set to and maintained in their current state, although the details of the output behavior vary greatly depending on the setting of the **Update I/O while in stop** option and the actions commanded via configured fieldbusses. Refer to Controller States Description (*see page 46*) for more details on these variations.

Set All Outputs to Default

Select this option by choosing **Set all outputs to default** in the **Behavior for outputs in Stop** drop-down menu of the **PLC settings** subtab of the **Controller Editor**. To access the **Controller Editor**, right-click on the controller in the device tree and select **Edit Object**.

This output behavior applies when the application is going from RUN state to STOPPED state or if the application is going from RUN state to HALT state. It also applies to CAN bus in the HALT controller state. Outputs are set to and maintained in their current state, although the details of the output behavior vary greatly depending on the setting of the **Update I/O while in stop** option and the actions commanded via configured fieldbusses. Refer to Controller States Description (*see page 46*) for more details on these variations.

Hardware Initialization Values

This output state applies in the BOOTING, EMPTY (following power cycle with no boot application or after the detection of a system error), and INVALID_OS states.

In the initialization state, analog, transistor, and relay outputs assume the following values:

- For an analog output: Z (high impedance)
- For a fast transistor output: Z (high impedance)
- For a regular transistor output: 0 Vdc
- For a 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 reset warm or cold.

The software **Initialization Values** are the initialization values of outputs images (%I, %Q, or variables mapped on %I or %Q).

By default, they are set to 0 but it is possible to map the I/O in a GVL and assign to the outputs a value different from 0.

Output Forcing

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

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

To do so, use the **Force values** command in the **Debug** menu.

Output forcing overrides all other commands to an output irrespective of the task programming that is being executed (this does not apply to an output currently used by the embedded controller functions, such as HSC/PTO/PWM).

When you logout of SoMachine when output forcing has been defined, you are presented with the option to retain output forcing settings. If you select this option, the output forcing continues to control the state of the selected outputs until you download an application or use one of the Reset commands.

When the option **Update I/O while in stop** is checked (default state), the forced outputs keep the forcing value even when the logic controller is in STOP.

Output Forcing Considerations

The output you wish to force must be contained in a task that is currently being executed by the controller. Forcing outputs in unexecuted tasks, or in tasks whose execution is delayed either by priorities or events will have no effect on the output. However, once the task that had been delayed is executed, the forcing will take effect at that time.

Depending on task execution, the forcing could impact your application in ways that may not be obvious to you. For example, an event task could turn on an output. Later, you may attempt to turn off that output but the event is not being triggered at the time. This would have the effect of the forcing being apparently ignored. Further, at a later time, the event could trigger the task at which point the forcing would take effect.

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 without removing the forcing.

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

Commanding State Transitions

Run Command

Effect: Commands a transition to the RUNNING controller state.

Starting Conditions: BOOTING or STOPPED state.

Methods for Issuing a Run Command:

- Run/Stop Input: If configured, command a rising edge to the Run/Stop input (assuming the Run/Stop switch is in the RUN position). Set the Run/Stop to 1 for all of the subsequent options to be effective.
Refer to Run/Stop Input (*see Magelis SCU, HMI Controller, Hardware Guide*) for more information.
- SoMachine Online Menu: Select the **Start** command.
- By an HMI command using the PLC_W. q_wPLCCControl and PLC_W. q_uiOpenPLCCControl system variables of the PLCSystem library (*see Magelis SCU, HMI Controller, PLCSystem Library Guide*).
- **Login with online change** option: An online change (partial download) initiated while the controller is in the RUNNING state returns the controller to the RUNNING state if successful.
- **Multiple Download Command:** sets the controllers into the RUNNING state if the **Start all applications after download or online change** option is selected, irrespective of whether the targeted controllers were initially in the RUNNING, STOPPED, HALT, or EMPTY state.
- The controller is restarted into the RUNNING state automatically under certain conditions.

Refer to Controller State Diagram (*see page 42*) for further details.

Stop Command

Effect: Commands a transition to the STOPPED controller state.

Starting Conditions: BOOTING, EMPTY, or RUNNING state.

Methods for Issuing a Stop Command:

- Run/Stop Input: If configured, command a value of 0 to the Run/Stop input. Refer to Run/Stop Input (*see Magelis SCU, HMI Controller, Hardware Guide*) for more information.
- SoMachine Online Menu: Select the **Stop** command.
- By an internal call by the application or an HMI command using the PLC_W. q_wPLCCControl and PLC_W. q_uiOpenPLCCControl system variables of the PLCSystem library (*see Magelis SCU, HMI Controller, PLCSystem Library Guide*).
- **Login with online change** option: An online change (partial download) initiated while the controller is in the STOPPED state returns the controller to the STOPPED state if successful.
- **Download Command:** implicitly sets the controller into the STOPPED state.
- **Multiple Download Command:** sets the controllers into the STOPPED state if the **Start all applications after download or online change** option is not selected, irrespective of whether the targeted controllers were initially in the RUNNING, STOPPED, HALT, or EMPTY state.

- REBOOT by USB file system download: The application download from a USB memory key will issue a REBOOT as its final command. The controller will be rebooted into the STOPPED state provided the other conditions of the boot sequence allow this to occur. Refer to *Saving your Application and Firmware on a USB Memory Key (see page 104)* and *Reboot (see page 103)* for further details.
- The controller is restarted into the STOPPED state automatically under certain conditions.

Refer to Controller State Diagram (*see page 42*) for further details.

Reset Warm

Effect: Resets all variables, except for the remanent variables, to their default values. Places the controller into the STOPPED state.

Starting Conditions:

- RUNNING, STOPPED, or HALT states.
- ControllerLockout = 0.

Methods for Issuing a Reset Warm Command:

- SoMachine Online Menu: Select the **Reset warm** command.
- By an internal call by the application or an HMI command using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the PLCSystem library (*see Magelis SCU, HMI Controller, PLCSystem Library Guide*).

Effects of the Reset Warm Command:

1. The application stops.
2. Forcing is erased.
3. Diagnostic indications for detected errors are reset.
4. The values of the retain variables are maintained.
5. The values of the retain-persistent variables are maintained.
6. All non-located and non-remanent variables are reset to their initialization values.
7. The values of %MW0 to %MW65535 registers are maintained.
8. All fieldbus communications are stopped and then restarted after the reset is complete.
9. All I/O are reset to their initialization values.

For details on variables, refer to Remanent Variables (*see page 59*).

Reset Cold

Effect: Resets all variables, except for the retain-persistent type of remanent variables, to their initialization values. Places the controller into the STOPPED state.

Starting Conditions:

- RUNNING, STOPPED, or HALT states.
- ControllerLockout = 0.

Methods for Issuing a Reset Cold Command:

- SoMachine Online Menu: Select the **Reset cold** command.
- By an internal call by the application or an HMI command using the PLC_W.q_wPLCCControl and PLC_W.q_uiOpenPLCCControl system variables of the PLCSystem library (*see Magelis SCU, HMI Controller, PLCSystem Library Guide*).

Effects of the Reset Cold Command:

1. The application stops.
2. Forcing is erased.
3. Diagnostic indications for detected errors are reset.
4. The values of the retain variables are reset to their initialization value.
5. The values of the retain-persistent variables are maintained.
6. All non-located and non-remanent variables are reset to their initialization values.
7. The values of %MW0 to %MW65535 registers are maintained.
8. All fieldbus communications are stopped and then restarted after the reset is complete.
9. All I/O are reset to their initialization values.

For details on variables, refer to Remanent Variables (*see page 59*).

Reset Origin

Effect: Resets all variables, including the remanent variables, to their initialization values. Erases all user files on the controller. Places the controller into the EMPTY state.

Starting Conditions:

- RUNNING, STOPPED, or HALT states.
- ControllerLockout = 0.

Methods for Issuing a Reset Origin Command:

- SoMachine Online Menu: Select the **Reset origin** command.

Effects of the Reset Origin Command:

1. The application stops.
2. Forcing is erased.
3. All user files (Boot application, data logging) are erased.
4. Diagnostic indications for detected errors are reset.
5. The values of the retain variables are reset.
6. The values of the retain-persistent variables are reset.
7. All non-located and non-remanent variables are reset.
8. All fieldbus communications are stopped.
9. Embedded Expert I/O are reset to their previous user-configured default values.
10. All other I/O are reset to their initialization values.

For details on variables, refer to Remanent Variables (*see page 59*).

Reboot

Effect: Commands a reboot of the controller.

Starting Conditions:

- ControllerLockout = 0.

Methods for Issuing the Reboot Command:

- Power cycle.
- REBOOT by USB file system download: The file application download from on a USB memory key will issue a REBOOT as its final command. The controller will be rebooted into the STOPPED state provided the other conditions of the boot sequence allow this to occur. Refer to Saving your Application and Firmware on a USB Memory Key (*see page 104*) for further details.

Effects of the Reboot:

1. The state of the controller depends on a number of conditions:
 - a. The controller state will be RUNNING if:
 - The Reboot was provoked by a power cycle, and
 - Controller state was RUNNING prior to the power cycle.
 - b. The controller state will be STOPPED if:
 - The Reboot was provoked by a Reboot by script, or
 - The boot application is different than the application loaded before the reboot, or
 - Controller state was STOPPED prior to a power cycle, or
 - The previously saved context is invalid.
 - c. The controller state will be EMPTY if:
 - There is no boot application or the boot application is invalid, or
 - d. The controller state will be INVALID_OS if there is no valid OS.
2. Forcing is maintained if the boot application is loaded successfully. If not, forcing is erased.
3. Diagnostic indications for detected errors are reset.
4. The values of the retain variables are restored if saved context is valid.
5. The values of the retain-persistent variables are restored if saved context is valid.
6. All non-located and non-remanent variables are reset to their initialization values.
7. All fieldbus communications are stopped and restarted after the boot application is loaded successfully.
8. All I/O are reset to their initialization values and then to their user-configured default values if the controller assumes a STOPPED state after the reboot.

For details on variables, refer to Remanent Variables (*see page 59*).

NOTE: The Check context test concludes that the context is valid when the application and the remanent variables are the same as defined in the Boot application.

NOTE: If you make an online change to your application program while your controller is in the RUNNING or STOPPED state but do not manually update your Boot application, the controller will detect a difference in context at the next reboot, the remanent variables will be reset as per a Reset cold command, and the controller will enter the STOPPED state.

Download Application

Effect: Loads your application executable into the RAM memory. Optionally, create a Boot application in the Flash memory.

Starting Conditions:

- RUNNING, STOPPED, HALT, and EMPTY states.
- ControllerLockout = 0.

Methods for Issuing the Download Application Command:

- SoMachine:
 - Two options exist for downloading a full application:
 - Download command.
 - Multiple Download command.

For important information on the application download commands, refer to Controller State Diagram ([see page 42](#)).

- USB memory key: Load Boot application file using a USB memory key connected to the controller USB host port. The updated file is applied at the next reboot. Refer to Saving your Application and Firmware on a USB Memory Key ([see page 104](#)) for further details.

Effects of the SoMachine Download Command:

1. The existing application stops and then is erased.
2. If valid, the new application is loaded and the controller assumes a STOPPED state.
3. Forcing is erased.
4. Diagnostic indications for detected errors are reset.
5. The values of the retain variables are reset to their initialization values.
6. The values of any existing retain-persistent variables are maintained.
7. All non-located and non-remanent variables are reset to their initialization values.
8. All fieldbus communications are stopped and then any configured fieldbus of the new application is started after the download is complete.
9. Embedded Expert I/O are reset to their previous user-configured default values and then set to the new user-configured default values after the download is complete.
10. All other I/O are reset to their initialization values and then set to the new user-configured default values after the download is complete.

For details on variables, refer to Remanent Variables ([see page 59](#)).

Effects of USB memory key Download Command:

There is no effect until the next reboot. At the next reboot, the effects are the same as a reboot with an invalid context. Refer to Reboot ([see page 103](#)).

Error Detection, Types, and Management

Error Management

The controller detects and manages three types of errors:

- external errors
- application errors
- system errors

This table describes the types of errors that may be detected:

Type of Error Detected	Description	Resulting Controller State
External Error	External errors are detected by the system while RUNNING or STOPPED but do not affect the ongoing controller state. An external error is detected in the following cases: <ul style="list-style-type: none"> ● A connected device reports an error to the controller. ● The controller detects an error with an external device, for example, when the external device is communicating but not properly configured for use with the controller. ● The controller detects an error with the state of an output. ● The controller detects a communication interruption with a device. ● The boot application in Flash memory is not the same as the one in RAM. 	RUNNING with External Error Detected Or STOPPED with External Error Detected
Application Error	An application error is detected when improper programming is encountered or when a task watchdog threshold is exceeded.	HALT
System Error	A system error is detected when the controller enters a condition that cannot be managed during runtime. Most such conditions result from firmware or hardware exceptions, but there are some cases when incorrect programming can result in the detection of a system error, for example, when attempting to write to memory that was reserved during runtime, or when a system watchdog time-out occurs. NOTE: There are some system errors that can be managed by runtime and are therefore treated like application errors.	BOOTING → EMPTY

NOTE: Refer to the HMI SCU PLCSystem Library Guide (*see Magelis SCU, HMI Controller, PLCSystem Library Guide*) for more detailed information on diagnostics.

Remanent Variables

Overview

Remanent variables can retain their values in the event of power outages, reboots, resets, and application program downloads. There are 2 types of remanent variables, declared individually as **Retain** or **Persistent retain**. **Retain** variables can be declared within a **POU** or as **Global**. **Persistent retain** variables, however, are only declarable as **Global**.


Remanent variables are retained only if the battery charge is sufficient.

This table describes the behavior of remanent variables in each case:

Action	VAR	VAR RETAIN	VAR GLOBAL PERSISTENT RETAIN
Online change to application program	X	X	X
Stop	X	X	X
Power cycle	-	X	X
Reset warm	-	X	X
Reset cold	-	-	X
Reset origin	-	-	-
Download of application program	-	-	X
X The value is maintained - The value is reinitialized			

Adding Retain Persistent Variables

Declare retain persistent (**VAR GLOBAL PERSISTENT RETAIN**) symbols in the **PersistentVars** window:

Step	Action
1	Select the Application node in the Applications tree .
2	Click  .
3	Choose Add other objects → Persistent variables
4	Click Add . Result: The PersistentVars window is displayed.

Chapter 8

Controller Configuration

Introduction

This chapter describes how to configure the Magelis SCU HMI Controller.

What Is in This Chapter?

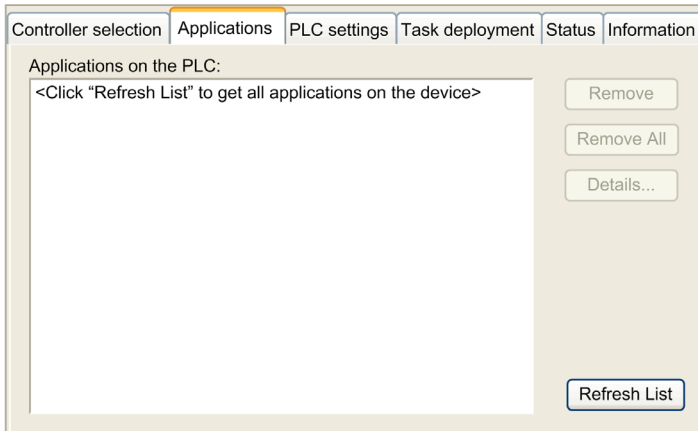
This chapter contains the following topics:

Topic	Page
Controller Parameters	62
Controller Selection	63
Applications	64
PLC Settings	65

Controller Parameters

Controller Parameters

To open the device editor, double-click **HMISCUxx5** in the Devices tree (*see page 16*):



Tabs Description

Tab	Description	Restriction
Controller selection (<i>see page 63</i>)	Manages the connection from PC to the controller: <ul style="list-style-type: none"> ● helping you find a controller in a network, ● presenting the list of available controllers, so you can connect to the selected controller and manage the application in the controller, ● helping you physically identify the controller from the device editor, ● helping you change the communication settings of the controller. 	Online mode only
Applications	Presents the application running on the controller and allows removing the application from the controller.	Online mode only
PLC settings (<i>see page 65</i>)	Configuration of: <ul style="list-style-type: none"> ● application name ● I/O behavior in stop ● bus cycle options 	–
Task deployment	Displays a list of I/Os and their assignments to tasks.	After compilation only
Status	Displays device-specific status and diagnostic messages.	–
Information	Displays general information about the device (name, description, provider, version, image).	–

Controller Selection

Introduction

This tab allows you to manage the connection from the PC to the controller:

- Helping you find a controller in a network.
- Presenting the list of controllers, so you can connect to the selected controller and manage the application inside the controller.
- Helping you physically identify the controller from the device editor.
- Helping you change the communication settings of the controller.

Process Communication Settings

The **Process communication settings** window lets you change the Ethernet communication settings. To do so, click **Controller selection** tab. The list of controllers available in the network appears. Select and right-click the required row and click **Process communication settings ...** in the context menu.

You can configure the Ethernet settings in the **Process communication settings** window in 2 ways:

- Without the **Save settings permanently** option:
Configure the communication parameters and click **OK**. These settings are immediately taken into account and are not kept if the controller is reset. For the next resets, the communication parameters configured into the application are taken into account.
- With the **Save settings permanently** option:
You can also activate the **Save settings permanently** option before you click **OK**. Once this option is activated, the Ethernet parameters configured here are always taken into account on reset instead of the Ethernet parameters configured into the SoMachine application.

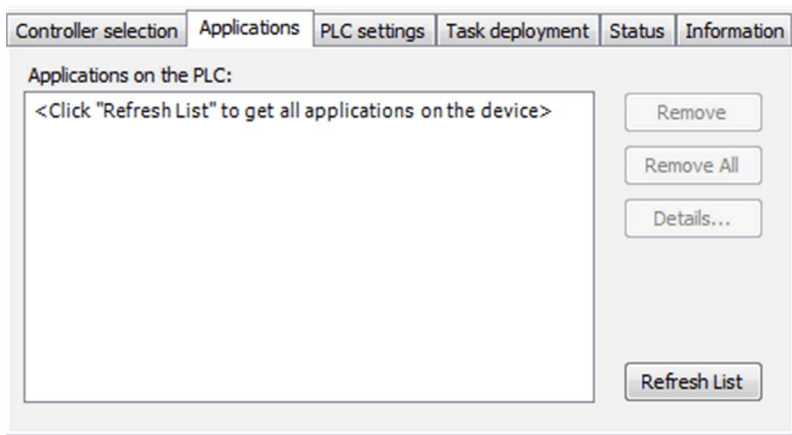
For more information on the **Controller selection** view of the device editor, refer to the SoMachine Programming Guide.

Applications

Overview

The **Applications** view of the device editor serves to scan and to remove applications on the controller. Information on the content of the application can be available as well as some details on the application properties.

Applications view of the device editor:



The **Applications** view provides the following elements:

Element		Description
Applications on the PLC		List of the names of applications, which have been found on the controller during the last scan. NOTE: SoMachine controllers currently support only one application in a device at a time.
Buttons	Remove	The application currently selected in the list is removed from the controller.
	Remove all	All applications are removed from the controller.
	Refresh List	The controller is scanned for applications and the list is updated.

PLC Settings

Overview

The figure below presents the **PLC Settings** tab:

Element	Description
Application for I/O handling	By default, set to Application because there is only one application in the controller.
PLC settings	Update IO while in stop If this option is activated (default), the values of the input and output channels get also updated when the controller is stopped.
	Behavior for outputs in Stop From the selection list, choose one of the following options to configure how the values at the output channels should be handled in case of Controller stop: <ul style="list-style-type: none"> ● Keep current values: The current values will not be modified. ● Set all outputs to default: The default (fallback) values resulting from the mapping will be assigned. NOTE: This option is not taken into account for the outputs used by the HSC, PTO, or PWM.
	Update all variables in all devices If this option is activated, then for all devices of the current controller configuration all I/O variables will get updated in each cycle of the bus cycle task. This corresponds to the option Always update variables , which can be set separately for each device in the I/O Mapping dialog.
Bus cycle options	Bus cycle task This configuration setting is the parent for all Bus cycle task parameters used in the application device tree. Some devices with cyclic calls, such as a CANopen manager , can be attached to a specific task. In the device, when this setting is set to Use parent bus cycle setting , the setting set for the controller is used. The selection list offers all tasks currently defined in the active application. The default setting is the MAST task. NOTE: <unspecified> means that the task is in "slowest cyclic task" mode.

Element		Description
Additional settings	Generate force variables for IO mapping	Not used.
	Enable Diagnosis for devices	Not used.

Chapter 9

Embedded Functions

Overview

This chapter describes how to configure the embedded functions of the Magelis SCU HMI Controller.

The number of inputs and outputs dedicated to the embedded function depends on the HMI controller reference (*see page 13*).

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
I/O Embedded Function	68
HSC Embedded Function	72
PTO_PWM Embedded Function	74
Analog I/O Embedded Function	77
Analog Temperature Embedded Function	82

I/O Embedded Function

Overview

The embedded I/O function allows configuring of the controller inputs.

The table describes the digital inputs of the controllers:

Reference	Digital Input	Digital Output
HMISCU6A5	14 regular inputs and 2 fast inputs (HSC)	8 regular outputs and 2 fast outputs (PTO) ⁽¹⁾
HMISCU8A5		
HMISAC		
HMISCU6B5	6 regular inputs and 2 fast inputs (HSC)	6 regular outputs and 2 fast outputs (PTO) ⁽¹⁾
HMISCU8B5		
HMI SBC		
1 The fast outputs can be used either as regular outputs or as fast outputs for Pulse Train Output (PTO), Pulse Width Modulation (PWM) functions, or reflex output for high speed counter (HSC).		

Accessing the I/O Configuration Window

Follow these steps to access the embedded I/O configuration window:

Step	Description
1	Click HMISCUxx5 → Embedded Functions → IO in the Devices tree .
2	Click the I/O Configuration tab.

I/O Configuration Window

The window allows you to configure the embedded digital inputs:

The screenshot shows the 'I/O Configuration' window with the 'I/O Mapping' tab selected. The window displays a table of parameters for digital inputs. The parameters are organized into a tree view on the left, with a table on the right showing the configuration for each parameter. The 'Filter' parameter for FI1 is highlighted in blue.

Parameter	Type	Value	Default Value	Unit
Inputs				
FI0				
Filter	Enumeration of BYTE	No	No	ms
Latch	Enumeration of BYTE	No	No	
Event	Enumeration of BYTE	No	No	
Bounce Filter	Enumeration of BYTE	4	4	us
Run/Stop	Enumeration of BYTE	No	No	
FI1				
Filter	Enumeration of BYTE	No	No	ms
Latch	Enumeration of BYTE	No	No	
Event	Enumeration of BYTE	No	No	
Bounce Filter	Enumeration of BYTE	4	4	us
Run/Stop	Enumeration of BYTE	No	No	
DI0				
Filter	Enumeration of BYTE	No	No	ms
Latch	Enumeration of BYTE	No	No	
Event	Enumeration of BYTE	No	No	
Bounce Filter	Enumeration of BYTE	4	4	us
Run/Stop	Enumeration of BYTE	No	No	
DI1				
Filter	Enumeration of BYTE	No	No	ms
Latch	Enumeration of BYTE	No	No	
Event	Enumeration of BYTE	No	No	
Bounce Filter	Enumeration of BYTE	4	4	us
Run/Stop	Enumeration of BYTE	No	No	

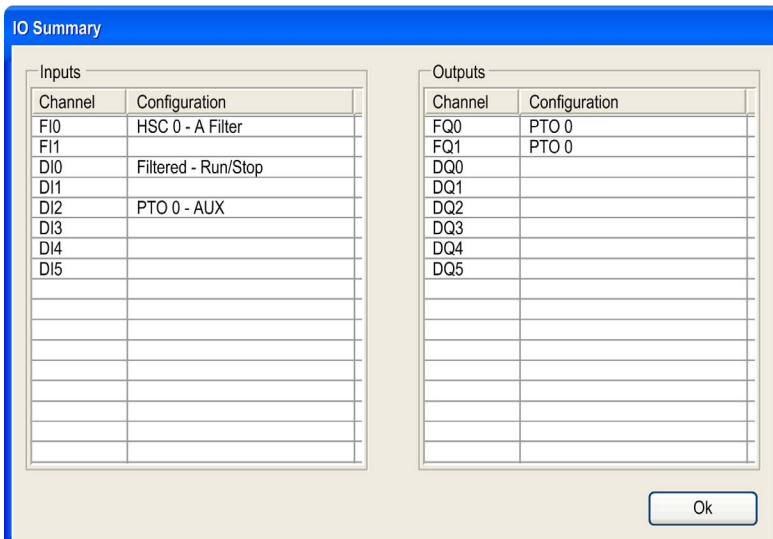
Filtering value reduces the effect of noise on a controller input

IO Summarize...

NOTE: If the selection is gray, the parameter is unavailable.

NOTE: For more information on the I/O Mapping tab, refer to the CoDeSys online help in SoMachine.

When you click the **IO Summarize** button, the **IO Summary** window appears. It allows you to check your configured I/O mapping:



Configuration Parameters

For each digital input, you can configure the parameters:

Parameter	Value	Description	Constraint
Filter	No* 3 ms 12 ms	Reduces the effect of noise on a controller input.	Available if Latch and Event are disabled. In the other cases, this parameter is disabled and its value is No.
Latch	No* Yes	Allows incoming pulses with amplitude widths shorter than the controller scan time to be captured and recorded.	This parameter is only available for fast inputs FIO and FI1 . Available if Run/Stop is disabled.
Event	No* Rising edge Falling edge Both edges	Event Detection for the purpose of triggering an External Task.	This parameter is only available for fast inputs FIO and FI1 . Available if both Latch and Run/Stop are disabled.
Bounce Filter	4 μs* 40 μs	Reduces the effect of bounce on a controller input.	Available if Latch is enabled or Event is enabled. In the other cases, this parameter is disabled and its value is 4 μs.

Parameter	Value	Description	Constraint
Run/Stop	No* Yes	The Run/Stop input can be used to run or stop a program in the controller.	Any input can be configured as Run/Stop , but only one input can be configured for this purpose.
* parameter default value			

NOTE: The parameter is unavailable if the selection displays as gray in color.

Run/Stop Input

This table presents the different states:

Input states	Result
State 0	Stops the controller and ignores external Run commands.
A rising edge	From the STOPPED state, initiate a start-up of an application in RUNNING state.
State 1	The application can be controlled by: <ul style="list-style-type: none"> ● SoMachine (Run/Stop) ● application (Controller command) ● network command (Run/Stop command)

NOTE: Run/Stop input is managed even if the option **Update I/O while in stop** is not selected in Controller Device Editor (**PLC settings** tab) (*see page 65*).

Inputs assigned to configured expert functions cannot be configured as Run/Stop inputs.

For further details about controller states and states transitions, refer to Controller State Diagram (*see page 42*).

⚠ WARNING
<p>UNINTENDED MACHINE OR PROCESS START-UP</p> <ul style="list-style-type: none"> ● Verify the state of security of your machine or process environment before applying power to the Run/Stop input. ● Use the Run/Stop input to help prevent the unintentional start-up from a remote location. <p>Failure to follow these instructions can result in death, serious injury, or equipment damage.</p>

HSC Embedded Function

Overview

The HSC function can execute fast counts of pulses from sensors, encoders, switches, and so on, that are connected to the dedicated fast inputs.

There are 2 types of HSC:

- **Simple** type: a single input counter.
- **Main** type: a counter that uses up to 4 inputs (2 fast inputs and 2 standard inputs) and 2 reflex outputs.

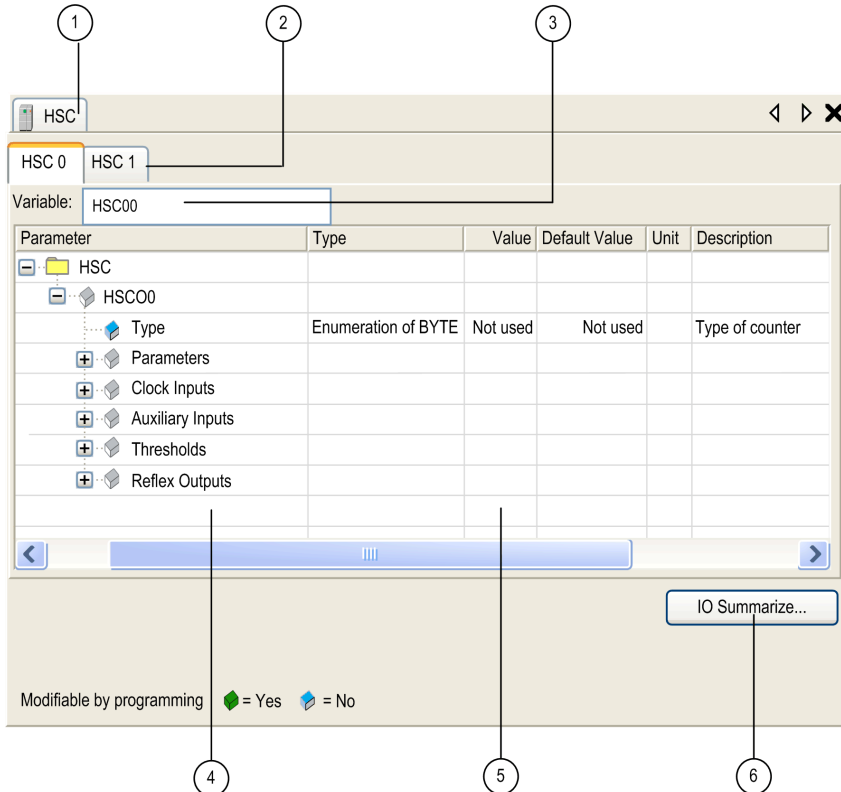
Accessing the HSC Configuration Window

Follow these steps to access the embedded HSC configuration window:

Step	Description
1	In the Devices tree , double-click HMISCUxx5 → Embedded Functions → HSC . Result: The HSC Configuration window is displayed.

HSC Configuration Window

The figure shows a sample HSC configuration window used to configure the HSC:



The table describes the areas of the HSC configuration window:

Number	Action
1	If necessary, select the HSC tab to access the HSC configuration Windows.
2	Select a specific HSC • tab to access the HSC channel you need to configure.
3	Choose the type of HSC (Simple or Main) you want. The global variable name representing the channel instance can be defined here. Default for HSC 0 is HSC00 , and for HSC 1 is HSC01 .
4	Expand each parameter by clicking the plus sign next to it to access its settings.
5	Configuration window where the HSC parameters are set depending on the mode used.
6	When you click the IO Summarize button, the IO Summary window appears. It allows you to check your configured physical I/O mapping.

For detailed information on configuration parameters, refer to HMI SCU HSC choice matrix (see *Magelis SCU, HMI Controller, HSC Library Guide*).

PTO_PWM Embedded Function

Overview

The PTO embedded function can provide 2 different functions:

PTO The PTO (Pulse Train Output) implements digital technology that provides precise positioning for open loop control of motor drives.

PWM The PWM (Pulse Width Modulation) function generates a programmable square wave signal on a dedicated output (*see Magelis SCU, HMI Controller, PTO/PWM Library Guide*) with adjustable duty cycle and frequency.

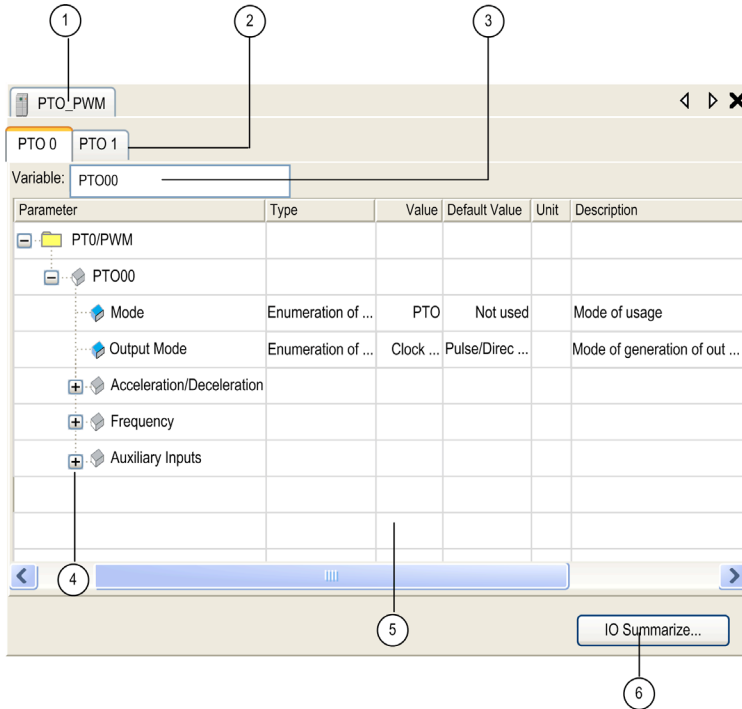
Accessing the PTO_PWM Configuration Window

Follow this step to access the PTO_PWM embedded function configuration window:

Step	Description
1	In the Devices tree , double-click HMISCUxx5 → Embedded Functions → PTO_PWM . Result: The PTO_PWM window is displayed.

PTO_PWM Configuration Window

The figure shows a sample PTO_PWM configuration window used to configure a PTO or PWM:



The table describes the areas of the PTO_PWM configuration window:

Number	Action
1	If necessary, select the PTO_PWM tab to access the PTO_PWM configuration Windows.
2	Select a specific PTO tab to access the PTO_PWM channel to configure.
3	Choose the type of PTO_PWM, (PTO (default) or PWM). Use the field Variable to change the Global Variable name representing the instance of the channel. NOTE: The default variable name for PTO 0 channel is PTO00 . For PTO 1 channel, it is PTO01 .
4	You can expand each parameter by clicking the plus sign next to it to access its settings.
5	Configuration window where the embedded function is used for: <ul style="list-style-type: none"> ● PTO ● PWM
6	Click the IO Summarize button. Result: The IO Summary window appears that shows the configured I/O mapping.

For detailed information on configuration parameters, refer to:

- PTO configuration (*see Magelis SCU, HMI Controller, PTO/PWM Library Guide*).
- PWM configuration.

Analog I/O Embedded Function

Overview

The HMISCUxB5 HMI controllers have embedded analog I/O:

- 2 analog inputs
- 2 analog outputs

For information about the technical characteristics of the analog I/O, refer to the Hardware Guide (*see Magelis SCU, HMI Controller, Hardware Guide*).

Accessing the Analog I/O Configuration Window

Follow these steps to access the analog I/O embedded function configuration window:

Step	Description
1	Click HMISCUxx5 → Embedded Functions → Analog .
2	Select the I/O Configuration tab.

Analog I/O Configuration Window

This window allows you to configure the analog I/O:

Analog I/O Mapping		I/O Configuration			
Parameter	Type	Value	Default Value	Unit	Description
[-] Inputs					
[-] IW0					
Type	Enumeration of BYTE	-10...10 V	Not used		Range mode
Scope	Enumeration of BYTE	Normal	Not used		Unit
Minimum	INT	-8192	0		Minimum value
Maximum	INT	8191	8191		Maximum value
[-] IW1					
Type	Enumeration of BYTE	4..20 mA	Not used		Range mode
Scope	Enumeration of BYTE	Customized	Not used		Unit
Minimum	INT	-32768	0		Minimum value
Maximum	INT	32767	8191		Maximum value
[-] Outputs					
[-] QW0					
Type	Enumeration of BYTE	0...10 V	Not used		Range mode
Scope	Enumeration of BYTE	Normal	Not used		Unit
Minimum	INT	0	0		Minimum value
Maximum	INT	8191	4095		Maximum value
[-] QW1					
Type	Enumeration of BYTE	0...20 mA	Not used		Range mode
Scope	Enumeration of BYTE	Customized	Not used		Unit
Minimum	INT	-32768	0		Minimum value
Maximum	INT	32767	4095		Maximum value

NOTE: The parameter is unavailable if the selection displays as gray in color.

NOTE: Embedded analog I/Os are always physically updated by the MAST task.

I/O Configuration Tab

To configure the HMI SCU, select the **I/O Configuration** tab.

The table describes the analog parameters configuration:

Parameter		Value	Description	Constraint
Type		Not used * – 10...10 V 0...10 V 0...20 mA 4...20 mA	Range mode	–
Scope		Normal * Customized	Unit	Available if Type value is defined.
Minimum	Normal Analog input	– 10...10 V: – 8192 0...10 V: 0 0...20 mA: 0 4...20 mA: 0	Minimum value	Not configurable.
	Normal Analog output	– 10...10 V: – 2048 0...10 V: 0 0...20 mA: 0 4...20 mA: 0		Not configurable.
	Customized Analog I/O	– 10...10 V: – 32768 0...10 V: – 32768 0...20 mA: – 32768 4...20 mA: – 32768		Inferior that the configured maximum. Configured minimum must be less than the configured maximum.
Maximum	Normal Analog input	– 10...10 V: 8191 0...10 V: 16383 0...20 mA: 16383 4...20 mA: 16383	Maximum value	Not configurable.
	Normal Analog output	– 10...10 V: 2047 0...10 V: 4095 0...20 mA: 4095 4...20 mA: 4095		Not configurable.
	Customized Analog I/O	– 10...10 V: 32767 0...10 V: 32767 0...20 mA: 32767 4...20 mA: 32767		Superior that the configured minimum. Configured maximum must be more than the configured minimum.
* parameter default value				

If you have physically wired the analog channel for a voltage signal and you configure the channel for a current signal in SoMachine, you may damage the analog circuit.

NOTICE

INOPERABLE EQUIPMENT

Verify that the physical wiring of the analog circuit is compatible with the software configuration for the analog channel.

Failure to follow these instructions can result in equipment damage.

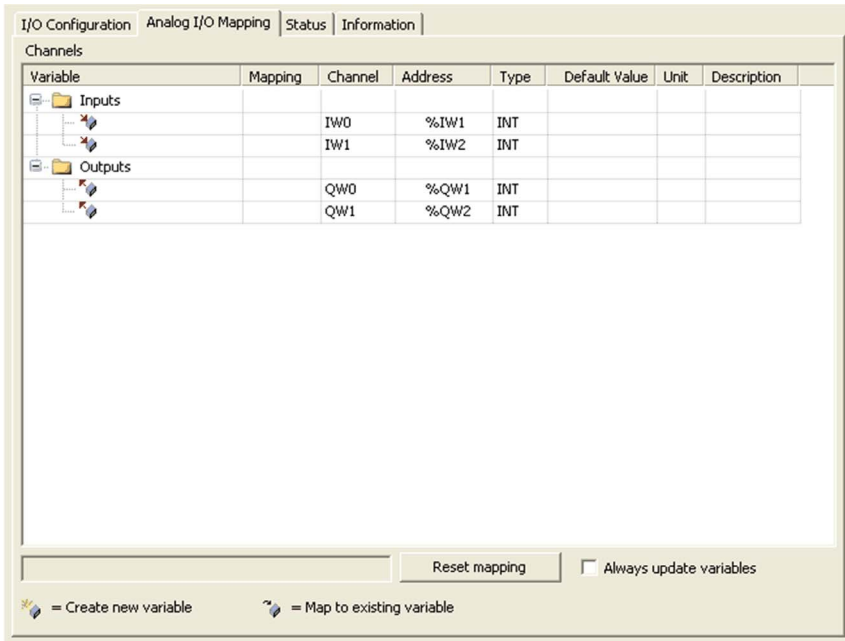
Analog to Digital Conversion Rate

Analog-to-digital conversion rate for data acquisition is made using the converter hardware. All analog I/O that are configured are converted every 2 ms.

I/O Mapping Tab

Variables can be defined and named in the **Analog I/O Mapping** tab. Additional information such as topological addressing is also provided in this tab.

This window shows the **Analog I/O Mapping** tab:



The table describes the I/O mapping configuration:

Variable	Channel	Type	Default value	Description
Inputs	IW0	INT	–	Current value of the input 0
	IW1			Current value of the input 1
Outputs	QW0	INT	–	Current value of the output 0
	QW1			Current value of the output 1

Analog Temperature Embedded Function

Overview

The HMISCUxB5 HMI controllers have embedded analog temperature with 2 inputs. The HMI SCU is designed for various sensor types. The sensor type must be specified because of the different adjustment values.

For information about the technical characteristics of the analog temperature inputs, refer to the Hardware Guide (*see Magelis SCU, HMI Controller, Hardware Guide*).

Access to Analog Temperature Configuration

Use this procedure to access temperature input configuration:

Step	Action
1	Click HMISCUxx5 → Embedded Functions → Temperature .
2	Select the I/O Configuration (<i>see page 82</i>) or Temperature I/O Mapping (<i>see page 86</i>) tab.

Analog Temperature Input Configuration Window

This window allows you to configure the analog temperature inputs:

I/O Configuration Temperature I/O Mapping Status Information						
Parameter	Type	Value	Default Value	Unit	Description	
[-] Inputs						
[-] IW0						
Type	Enumeration of BYTE	Thermocouple J	Not used		Range mode	
Scope	Enumeration of BYTE	Celsius (0.1 °C)	Not used		Unit	
Minimum	INT (-2000...7600)	-2000	-32768		Minimum value	
Maximum	INT (-2000...7600)	7600	32767		Maximum value	
[-] IW1						
Type	Enumeration of BYTE	PT100(3 Wire)	Not used		Range mode	
Scope	Enumeration of BYTE	Customized	Not used		Unit	
Minimum	INT (-32768...32767)	-32768	-32768		Minimum value	
Maximum	INT (-32768...32767)	32767	32767		Maximum value	

NOTE: Embedded analog I/Os are always physically updated by the MAST task.

I/O Configuration Tab

The table describes the analog input parameter configuration:

Parameter	Value	Description	Constraint
Type	Not used * Thermocouple J Thermocouple K Thermocouple R Thermocouple B Thermocouple S Thermocouple T Thermocouple E Thermocouple N PT100 (2/4 wire) PT1000 (2/4 wire) NI100 (2/4 wire) NI1000 (2/4 wire) PT100 (3 wire) PT1000 (3 wire) NI100 (3 wire) NI1000 (3 wire)	Range mode	–
Scope	Not used * Normal Customized Celsius (0.1 °C) Fahrenheit (0.1 °F)	Unit	Available if Type value is defined.

Parameter		Value	Description	Constraint
Minimum	Normal	0	Minimum value	Not configurable.
	Customized	- 32768	Minimum value	Lowest configurable value.
	Celsius	Thermocouple J: -2000 Thermocouple K: -2400 Thermocouple R: 0 Thermocouple B: 2000 Thermocouple S: 0 Thermocouple T: -2000 Thermocouple E: -2000 Thermocouple N: -2000 PT100: -2000 PT1000: -2000 NI100: -500 NI1000: -500	Minimum value	Not configurable.
	Fahrenheit	Thermocouple J: -3280 Thermocouple K: -4000 Thermocouple R: 32 Thermocouple B: -3920 Thermocouple S: 32 Thermocouple T: -3280 Thermocouple E: -3280 Thermocouple N: -3280 PT100: -3280 PT1000: -3280 NI100: -580 NI1000: -580	Minimum value	Not configurable.

Parameter		Value	Description	Constraint
Maximum	Normal	65535	Maximum value	Not configurable.
	Customized	32767	Maximum value	Highest configurable value.
	Celsius	Thermocouple J: 7600 Thermocouple K: 13700 Thermocouple R: 16000 Thermocouple B: 18000 Thermocouple S: 16000 Thermocouple T: 4000 Thermocouple E: 9000 Thermocouple N: 13000 PT100: 6000 PT1000: 6000 NI100: 2000 NI1000: 2000	Maximum value	Not configurable.
	Fahrenheit	Thermocouple J: 14000 Thermocouple K: 24980 Thermocouple R: 29120 Thermocouple B: 32720 Thermocouple S: 29120 Thermocouple T: 7520 Thermocouple E: 16520 Thermocouple N: 23720 PT100: 11120 PT1000: 11120 NI100: 3920 NI1000: 3920	Maximum value	Not configurable.
* parameter default value				

If you have physically wired the analog channel for a voltage signal and you configure the channel for a current signal in SoMachine, you may damage the analog circuit.

NOTICE

INOPERABLE EQUIPMENT

Verify that the physical wiring of the analog circuit is compatible with the software configuration for the analog channel.

Failure to follow these instructions can result in equipment damage.

NOTE:

- To reduce cycle time, do not activate a channel when there is no sensor connected.
- The data type is always INT which is a 16 bit integer value. The minimum and maximum values are shown according to the entries in the table above. (for example, for thermocouple J in Celsius the Type would read: **INT**(- 2000...7600))
- The parameter is unavailable if the selection displays as gray in color.

Analog to Digital Conversion Rate

Analog-to-digital conversion rate for data acquisition is made using the converter hardware. All temperature analog I/Os that are configured are converted every 2 ms.

Terminal Temperature (Cold Junction) Compensation

Terminal Temperature (Cold Junction) Compensation is measured and calculated internally. No extra temperature measurement of the terminal is required.

Temperature I/O Mapping Window

Variables can be defined and named in the **Temperature I/O Mapping** window. Additional information such as topological addressing is also provided in this tab.

The table describes the I/O mapping configuration:

Variable	Channel	Type	Default Value	Description
Inputs	IW0	INT	-	Current value of the input 0
	IW1			Current value of the input 1

Chapter 10

Ethernet Configuration

Introduction

This chapter describes how to configure the Ethernet network interface of the Magelis SCU HMI Controller

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
IP Address Configuration	88
Modbus TCP Client/Server	90

IP Address Configuration

Introduction

Setting up an Ethernet connection and IP address configuration with the HMI controllers is accomplished with Vijeo-Designer **Property Inspector** → **Download** → **Ethernet**.

There are two ways to assign the IP address of the controller with Vijeo-Designer:

- DHCP server
- Fixed IP address

The Ethernet configuration parameters are applied after a download of the HMI application.

For more information on how to configure the Ethernet connection between the computer and the HMI SCU controller, refer to Vijeo-Designer online help.

NOTE: If the above addressing modes are not operational, the HMI SCU controller starts with a default IP address derived from its MAC address.

Carefully manage the IP addresses because each device on the network requires a unique address. Having multiple devices with the same IP address can cause unintended operation of your network and associated equipment.

WARNING

UNINTENDED EQUIPMENT OPERATION

- Verify that there is only one master controller configured on the network or remote link.
- Verify that all devices have unique addresses.
- Obtain your IP address from your system administrator.
- Confirm that the IP address of the device is unique before placing the system into service.
- Do not assign the same IP address to any other equipment on the network.
- Update the IP address after cloning any application that includes Ethernet communications to a unique address.

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

NOTE: Verify that your system administrator maintains a record of all assigned IP addresses on the network and subnetwork, and inform the system administrator of all configuration changes performed.

Default IP Address

The default IP address is based on MAC address of the device. The first two bytes are 10 and 10. The last two bytes are the last two bytes of the MAC address of the device.

The default subnet mask is 255.0.0.0.

NOTE: A MAC address has a hexadecimal format and an IP address has a decimal format. Convert the MAC address into decimal format.

Example: If the MAC address is 00.80.F4.01.80.F2, the default IP address is 10.10.128.242.

Modbus TCP Client/Server

Introduction

Modbus TCP is based on a client/server model.

The Magelis SCU HMI Controller implements both client and server services so that it can initiate communications to other controllers and I/O devices, and respond to requests from other controllers, SCADA, HMIs, and other devices.

Modbus TCP Client

The Modbus TCP client supports the following function blocks from the PLCCommunication library without any configuration:

- ADDM
- READ_VAR
- SEND_RECV_MSG
- SINGLE_WRITE
- WRITE_READ_VAR
- WRITE_VAR

For further information, refer to the Function Block Descriptions (*see SoMachine, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide*).

Modbus TCP Server

The library `SE_Modbus_TCP_Slave` provides a communication function block, `ModbusServer`, for managing the communication between the Magelis SCU HMI Controller in the capacity of a Modbus server and any clients requesting Modbus services from the controller. For more details, refer to the appendix Library `SE_ModbusTCP_Slave` (*see page 125*).

Chapter 11

Serial Line Configuration

Introduction

This chapter describes how to configure the serial line communication of the Magelis SCU HMI Controller.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Serial Line Configuration	92
Serial Line Protocol Manager	95
SoMachine Network Manager	96
Modbus Manager	97

Serial Line Configuration

Introduction

The serial line **Configuration** window allows configuring the physical parameters of the serial line (baud rate, parity, and so on).

The Serial Line port(s) of your controller are configured for the SoMachine protocol by default when new or when you update the controller firmware. The SoMachine protocol is incompatible with other protocols such as that of Modbus Serial Line.

In an active Modbus configured serial line, if a new controller is connected or if controller firmware is updated, this can cause the other devices available on the serial line to stop communicating.

Check that the controller is not connected to an active Modbus serial line network before downloading a valid application having the concerned port or ports properly configured for the intended protocol.

WARNING

UNINTENDED EQUIPMENT OPERATION

Verify that your application has the Serial Line port(s) properly configured for Modbus before physically connecting the controller to an operational Modbus serial line network.

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

Access to Serial Line Configuration

To configure the serial line, proceed as follows:

Step	Action
1	Click HMISCUxx5 → COM1 in the Devices tree .
2	Click the Configuration tab.

Serial Line Configuration

The **Configuration** tab is displayed:

The following parameters must be identical for each serial device connected to the port:

Element	Description
Baud rate	Transmission speed in bits/s
Parity	Used for error detection
Data bits	Number of bits for transmitting data
Stop bits	Number of stop bits
Physical medium	Specify to use RS-232 or RS-485

The serial line ports of your controller are configured for the SoMachine protocol by default when new or when you update the controller firmware. The SoMachine protocol is incompatible with that of other protocols such as Modbus Serial Line. Connecting a new controller to, or updating the firmware of a controller connected to, an active Modbus configured serial line can cause the other devices on the serial line to stop communicating. Make sure that the controller is not connected to an active Modbus serial line network before first downloading a valid application having the concerned port or ports properly configured for the intended protocol.

NOTICE

INTERRUPTION OF SERIAL LINE COMMUNICATIONS

Be sure that your application has the serial line ports properly configured for Modbus before physically connecting the controller to an operational Modbus Serial Line network.

Failure to follow these instructions can result in equipment damage.

This table indicates the maximum baud rate value of the managers:

Manager	Maximum Baud Rate (Bits/S)
SoMachine Network Manager	115200
Modbus Manager	

Serial Line Protocol Manager

Overview

The Magelis SCU HMI Controller is equipped with 1 serial line:

Serial Line	Supported protocol
COM1	Modbus Manager (<i>see page 97</i>) SoMachine Network Manager (<i>see page 96</i>) (default)

The following table indicates the Manager baud rate characteristic:

Manager	Maximum Baud Rate (bits/s)	Supported Baud Rate (bits/s)
SoMachine Network Manager	115200	115200, 57600, 38400, 19200, 9600
Modbus Manager	115200	115200, 57600, 38400, 19200, 9600

NOTE: If you wish to use COM1 for protocols added in the Vijeo Designer editor, remove the **Modbus_Manager** and **SoMachine-Network_Manager** nodes from under the **COM1** node in the **Device Tree**. This allows the Vijeo Designer Runtime to take control of COM1.

SoMachine Network Manager

Introduction

The SoMachine Network Manager must be used if you want to exchange variables with a controller such as M241 or M251, or a SoMachine compatible HMI such as the Magelis GTO using SoMachine software protocol.

Adding the Manager

To add a **SoMachine-Network Manager** to your controller, click the **Plus Button**  next to the **COM1** node in the **Devices Tree**. In the **Add Device** window, select **SoMachine-Network Manager** and click the **Add Device** button.

Configuring the Manager


There is no configuration required for the SoMachine Network Manager.

Modbus Manager

Introduction

The **Modbus Manager** is used for Modbus RTU protocol in the master mode.

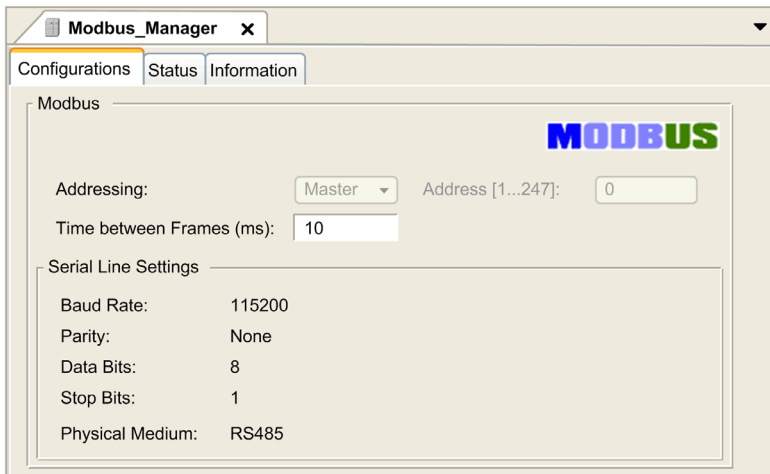
Adding the Manager

To add a **Modbus Manager** to your controller, click the **Plus Button**  next to the **COM1** node in the **Devices Tree**. In the **Add Device** window, select **Modbus Manager** and click the **Add Device** button.

Modbus Manager Configuration

To configure the Modbus Manager of your controller, double-click **Modbus Manager** in the **Devices tree**.

The Modbus Manager configuration window is displayed as below:



Set the parameters as described in the following table:

Element	Description
Addressing	The HMI SCU is configured as a master. No other options are available.
Address [1...247]	Modbus address of the device. For Modbus master, this is set to 0.
Time between Frames (ms)	Time to avoid bus-collision. Set this parameter to identical values for each Modbus device on the link.
Serial Line Settings	Parameters specified in the Serial Line configuration window (<i>see page 92</i>).

NOTE: On HMISCU, Modbus request function blocks from the PLC Communication Library require a specific **Time between Frames** based on the **Baud Rate** in order for communication at lower baud rates. Use the following formula to determine a suitable **Baud Rate/Time between Frames** combination:

Time between Frames (rounded up to the nearest millisecond) = 35000 / baud rate.

Modbus Master

When the controller is configured as a Modbus Master, the following function blocks are supported from the PLCCommunication Library:

- ADDM
- READ_VAR
- SINGLE_WRITE
- WRITE_READ_VAR
- WRITE_VAR

For further information, see Function Block Descriptions (*see SoMachine, Modbus and ASCII Read/Write Functions, PLCCommunication Library Guide*) of the PLCCommunication Library.

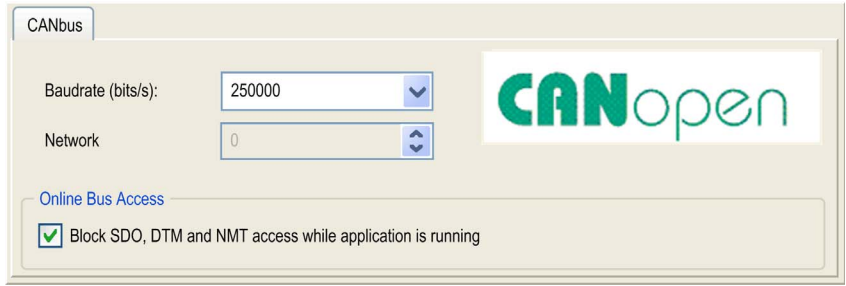
Chapter 12

CANopen Configuration

CANopen Interface Configuration

CAN Bus Configuration

To configure the **CAN** bus of your controller, proceed as follows:

Step	Action
1	In the Devices tree , double-click CAN .
2	Configure the baudrate (by default: 250000 bits/s):  NOTE: The Online Bus Access option allows you to block SDO, DTM, and NMT sending through the status screen.

When connecting a DTM to a device using the network, the DTM communicates in parallel with the running application. The overall performance of the system is impacted and can cause overload on the network.

WARNING


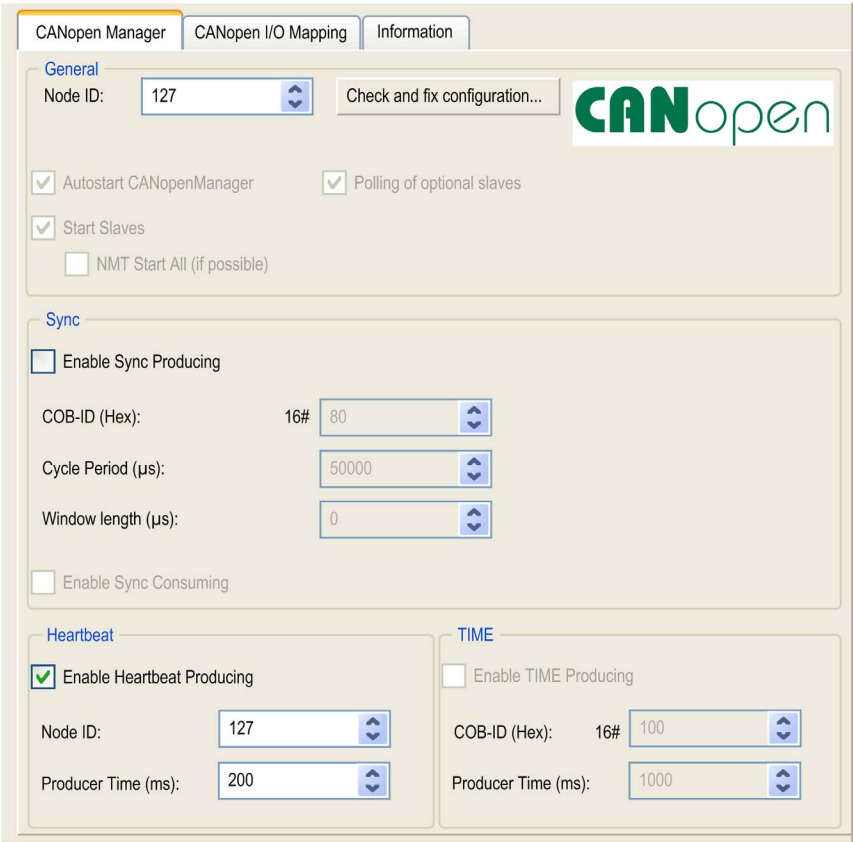
UNINTENDED EQUIPMENT OPERATION

You must consider the impact of DTM connections on the CANopen fieldbus load.

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

CANopen Manager Creation and Configuration

If the **CANopen Manager** is not already present below the **CAN** node, proceed as follows to create and configure it:

Step	Action
1	<p>To add a CANopen Optimized to your controller, click the Plus Button  next to the CAN node in the Devices Tree. In the Add Device window, select CANopen Optimized and click the Add Device button.</p> <p>For more information on adding a device to your project, refer to:</p> <ul style="list-style-type: none"> • Using the Drag-and-Drop Method (<i>see SoMachine, Programming Guide</i>) • Using the Contextual Menu or Plus button (<i>see SoMachine, Programming Guide</i>)
2	<p>Double-click CANopen_Optimized. Result: The CANopen Manager configuration window appears:</p> 

NOTE: If **Enable Sync Producing** is checked, the **CAN_x_Sync** task is added to the **Application → Task Configuration** node in the **Applications tree** tab.

Do not delete or change the **Type** or **External event** attributes of **CAN_x_Sync** tasks. If you do so, SoMachine will detect an error when you attempt to build the application, and you will not be able to download it to the controller.

If you uncheck the **Enable Sync Producing** option on the **CANopen Manager** subtab of the **CANopen_Optimized** tab, the **CAN0_Sync** task is automatically deleted from your program.

Adding a CANopen Device

Refer to the SoMachine Programming Guide for more information on Adding Communication Managers and Adding Slave Devices to a Communication Manager.

CANopen Operating Limits

The Magelis SCU HMI Controller CANopen master has the following operating limits:

Maximum number of slave devices	16
Maximum number of Received PDO (RPDO)	32
Maximum number of Transmitted PDO (TPDO)	32

WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not connect more than 16 CANopen slave devices to the controller.
- Program your application to use 32 or fewer Transmit PDO (TPDO).
- Program your application to use 32 or fewer Receive PDO (RPDO).

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

Chapter 13

Connecting the Magelis SCU HMI Controller to a PC

Connecting the Controller to a PC

Application Transfer

To transfer and run applications, connect your HMI SCU to a PC that has SoMachine installed. To transfer an application, use Ethernet, serial link or a USB memory key.

NOTICE

POSSIBLE ELECTRICAL DAMAGE TO CONTROLLER COMPONENTS

Connect the communication cable to the PC before connecting it to the controller.

Failure to follow these instructions can result in equipment damage.

NOTE: Only one HMI SCU can be connected to a computer at a time, except when using Ethernet.

Automatic Reboot after Application Transfer

The HMI SCU automatically reboots after a download of the application, this includes the control part (SoMachine) and the HMI part (Vijeo-Designer).

Firmware Update

When transferring an application (via Ethernet, USB download cable, or a USB memory key), the firmware update is done automatically (with a user prompt for confirmation) (*see page 105*).

Application Download with Firmware Downgrade

The HMI SCU can download an application and downgrade the firmware from a USB memory key. You must first save the application and the appropriate firmware version on a USB memory key.

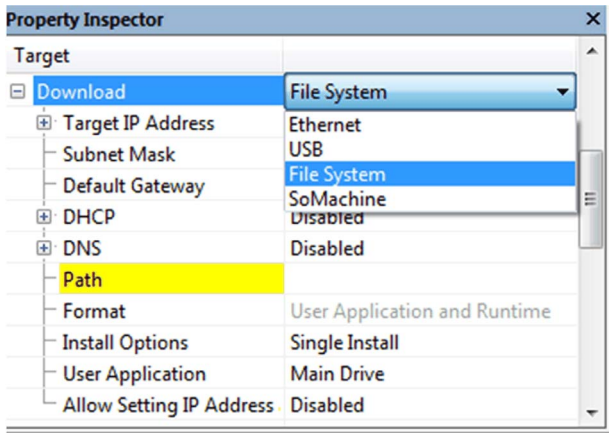
To download an application and downgrade the firmware of your controller follow the steps in the table below:

Step	Action
1	Turn off the power supply of your controller, prior to connecting the USB memory key.
2	Connect the USB memory key containing the application and firmware into the USB port of your controller.
3	Turn on your controller. Result: The application and the firmware version from the USB memory key are downloaded.

NOTE: If you plug a USB memory key containing the application and firmware while the controller is on, a screen is displayed asking you whether you want to install the application from the USB memory key.

Saving your Application and Firmware on a USB memory key

You can save your application and firmware on a FAT 32 USB memory key. To save, follow the steps in the table below:

Step	Action
1	Insert a USB memory key into the USB port of your computer.
2	Use the Quick Toolswitch menu (upper left in window) to select Vijeo-Designer . Result: The project switches to HMI and the main Vijeo-Designer window is displayed.
3	Right click on the controller node in the Navigator window, and select Properties . Result: The Property Inspector window is displayed.
4	Select File System from the Download menu as shown in the following figure:  <p>The screenshot shows the 'Property Inspector' window with a tree view on the left and a list of properties on the right. The 'Download' folder is expanded, and the 'File System' dropdown menu is open, showing options: Ethernet, USB, File System (highlighted), SoMachine, and Disabled. The 'Path' property in the tree view is highlighted in yellow.</p>
5	Set the directory from the Path menu to the USB memory key. NOTE: Select the root level of your USB memory key.
6	Click the OK button. Result: The directory is now set to the USB memory key.
7	Click Build → Download All from the Vijeo Designer main menu bar. Result: The application is saved onto the USB memory key.

NOTE: Use a FAT32 USB memory key to save your application and firmware.

Chapter 14

Firmware Update

Upgrading Magelis SCU HMI Controller Firmware/Runtime

Introduction

There are three methods of updating the Firmware/Runtime on an HMISCU:

- Use the **Runtime Installer Tool**. From the **Start Bar** in Windows, it is located in: **All Programs → Schneider Electric → Vijeo Designer 6.2 → Tools → Runtime Installer**.
- Download an HMI project from a newer version of **Vijeo Designer Editor** to the HMI SCU. This can be done via **Vijeo Designer Editor** or **SoMachine Editor** with any connection method. A prompt will appear if the Runtime version on the HMI is older than that of the **Vijeo Designer Editor** software.
- Download an HMI project from a newer version of **Vijeo Designer Editor** to a USB memory stick. Then insert the memory stick into the HMI SCU. A prompt will appear on the screen of the HMI SCU to install the new project.

For more information, refer to the **Vijeo Designer Online Help**, section Installing Vijeo Designer Runtime.

<h2><i>NOTICE</i></h2>
<p>LOSS OF APPLICATION DATA</p> <ul style="list-style-type: none">● Perform a backup of the application program to the hard disk of the PC before attempting a firmware change.● Restore the application program to the device after a successful firmware change. <p>Failure to follow these instructions can result in equipment damage.</p>

If 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.
- Do not place the device into service until the transfer has completed successfully.

Failure to follow these instructions can result in equipment damage.

The serial line ports of your controller are configured for the SoMachine protocol by default when new or when you update the controller firmware. The SoMachine protocol is incompatible with that of other protocols such as Modbus Serial Line. Connecting a new controller to, or updating the firmware of a controller connected to, an active Modbus configured serial line can cause the other devices on the serial line to stop communicating. Make sure that the controller is not connected to an active Modbus serial line network before first downloading a valid application having the concerned port or ports properly configured for the intended protocol.

NOTICE

INTERRUPTION OF SERIAL LINE COMMUNICATIONS

Be sure that your application has the serial line ports properly configured for Modbus before physically connecting the controller to an operational Modbus Serial Line network.

Failure to follow these instructions can result in equipment damage.

Chapter 15

Magelis SCU HMI Controller - Troubleshooting and FAQ

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Troubleshooting	108
Frequently Asked Questions	113

Troubleshooting

Introduction

This section lists the possible troubleshooting solutions with the HMI SCU, and procedures for troubleshooting them.

Transferring the Application is not Possible

Possible causes:

- PC cannot communicate with the controller.
- SoMachine not configured for the current connection.
- Is your application valid?
- Is the CoDeSys gateway running?
- Is the CoDeSys SP win running?

Resolution:

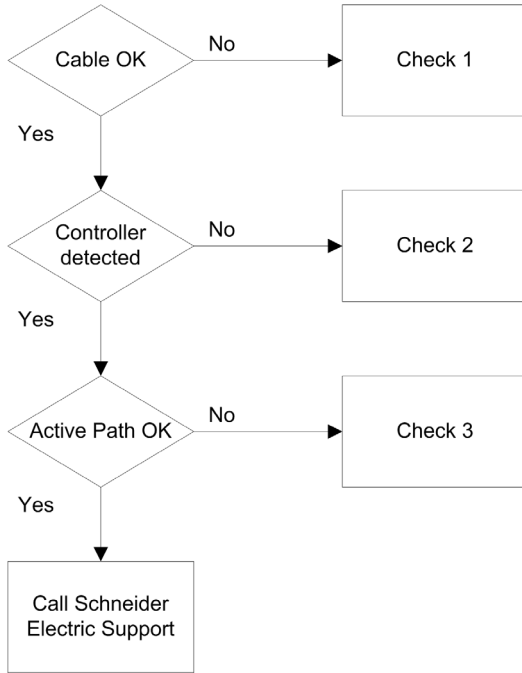
- Refer to Communication between SoMachine and the HMI SCU (*see page 108*).
- Your application program must be valid. Refer to the debugging section for more information.
- The CoDeSys gateway must be running:
 - a. click the CoDeSys Gateway icon in the task bar,
 - b. select **Start Gateway**.

Communication Between SoMachine and the HMI SCU is not Possible.

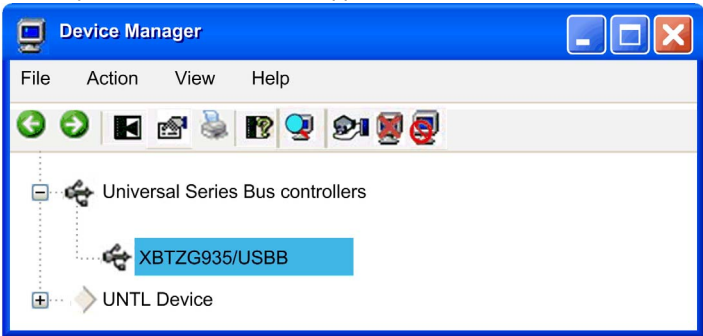

Possible causes:

- SoMachine not configured for the current connection.
- Incorrect cable usage.
- Controller not detected by the PC.
- Communication settings are not correct.
- The controller has detected an error or its firmware is invalid.

Resolution: Follow the flowchart below for troubleshooting purposes and then refer to the next table:



Check	Action
1	Verify that: <ul style="list-style-type: none"> ● The cable is correctly linked to the controller and to the PC, and is not damaged, ● You used the specific cable or adapter, depending on the connection type: <ul style="list-style-type: none"> ○ Ethernet and Serial link connection. ○ XBTZG935 cable for a USB connection. ○ XBTZG935 and XBTZGUSB or TCSXCNAMUM3P and XBTZGUSBB connection when the controller is mounted on a front panel.

Check	Action
2	<p>Verify that the HMI SCU has been detected by your PC:</p> <ol style="list-style-type: none"> 1. Click Start → Control Panel → System, then select the Hardware tab and click Device Manager, 2. Verify that the HMI SCU node appears in the list, as shown below:  <p>The screenshot shows the Windows Device Manager window. The 'Universal Series Bus controllers' category is expanded, and the device 'XBTZG935/USBB' is listed. The device name is highlighted with a blue selection box. Below it, 'UNTL Device' is also visible.</p> <ol style="list-style-type: none"> 3. If the HMI SCU node does not appear, or if there is an  icon in front of the node, disconnect and reconnect the cable on the controller side.
3	<p>Verify that the active path is correct:</p> <ol style="list-style-type: none"> 1. Double-click the controller node in the device view. 2. Verify that the HMI SCU node appears in bold, not in italic. <p>If not:</p> <ol style="list-style-type: none"> a. Stop the CoDeSys Gateway: right-click the icon in the task bar and select Stop Gateway. b. Disconnect and reconnect the cable on the controller side. c. Start the CoDeSys Gateway: right-click the icon in the task bar and select Start Gateway. d. Select the gateway in the controller window of SoMachine and click Scan network. Select the HMI SCU node and click Set active path. <p>NOTE: If your PC is connected to an Ethernet network, its address might have changed. In this case, the currently set active path is no longer correct and the HMI SCU node appears in italics. Select the HMI SCU node and click Resolve Name. If the node no longer appears in italics, click Set Active Path to correct this.</p>

Application Does Not Go to RUN State

Possible causes:

- No POU declared in the task.
- ControllerLockout activated.

Resolution:

As POUs are managed by tasks, add a POU to a task:

1. Double-click a task in the **Applications tree**.
2. Click the **Add Call** button in the task window.
3. Select the POU you want to execute in the **Input Assistant** window and click **OK**.
4. Unlock ControllerLockout in Vijeo Designer.

Creating the Boot Application is not Possible

Possible cause:

Operation not possible while the controller is in RUN state.

Resolution:

- Select **Stop Application**.
- Select **Create Boot Project**.

Changing Device Name does not work

Possible cause:

Application is running.

Resolution:

- Select **Stop Application**,
- Change device name.

CANopen Heartbeat is not sent on a regular basis

Possible cause:

Heartbeat value is not correct.

Resolution:

The Heartbeat of the CANopen master must be reset:

- Calculate the Heartbeat consumer time:
Heartbeat Consumer Time = Producer Time * 1.5
- Update the Heartbeat value

Monitoring of the POU is slow

Possible cause:

- Task interval is too small or POU is too big.
- Connection speed low between controller and device (over serial connection).

Resolution:

- Increase the configured task interval.
- Split the application into smaller POU's.

Out of Memory appears on the HMI screen

Possible cause:

- The number of variables and symbols shared between the controller and the HMI is too high.

Resolution:

- Decrease the number of variables and symbols shared between the controller and the HMI.
- Power cycle the HMI.

Frequently Asked Questions

What Programming Languages Are Supported by a HMI SCU?

These languages are supported:

- Continuous Function Chart (CFC)
- Function Block Diagram (FBD)
- Instruction List (IL)
- Ladder Logic Diagram (LD)
- Sequential Function Chart (SFC)
- Structured Text (ST)

What Variable Types Are Supported by an HMI SCU Controller?

Refer to the *Supported Variables* section.

Can I Use the SoMachine Network to Communicate with Equipment Connected to the Serial Line of my HMI SCU?

Communication is possible with an HMI SCU only if the serial line is configured with the *Network Protocol*.

Limitations:

- Slow access to the remote equipment.
- You cannot cascade other equipment.

For more information, refer to SoMachine - Network/Combo: HMI SCU part, available in the appendix of the Vijeo-Designer online help.

When Should I Use Freewheeling or Cyclic Mode?

Freewheeling or cyclic mode usage:

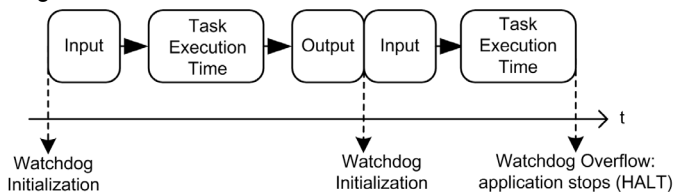
- Freewheeling: use this mode if you want a variable cycle time. The next cycle starts after a waiting period equal to 30% of the last cycle execution time.
- Cyclic: use this mode if you want to control the frequency cycle.

How Do I Configure the Watchdog?

You can configure the watchdog (control timer per task) using SoMachine by defining these parameters:

- **Time:** Set the maximum period of a given task. If the task execution time exceeds the maximum period, the watchdog is triggered.
- **Sensitivity:** Set the number of allowed consecutive and cumulate watchdog overruns before a watchdog trigger is generated.

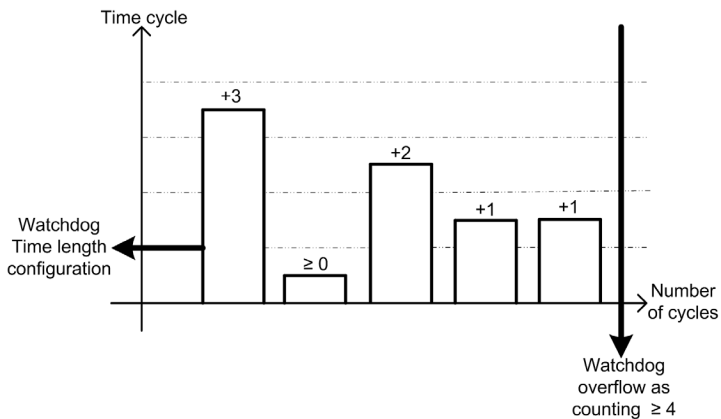
Depending on the **Time** and **Sensitivity** parameters, if the watchdog is triggered, the controller is stopped and goes into HALT mode. The associated task remains uncompleted, as shown in this diagram:



During a task execution, the firmware:

- Resets the overtime timer if the watchdog is not triggered
- Increments the overtime timer if the watchdog is triggered

In the following example the **Sensitivity** is set to 5:



What Does the Start all application after download or online change Checkbox Do?

- Case 1: Standalone HMI application download or HMI and Control applications download:
The BOOT state of the Control application is updated based on the checkbox setting
- Case 2: Control applications download only:
 - The setting of the checkbox takes effect after the download/online change.
 - The RUN of the control application at BOOT time is not affected.

Can I Connect Several HMI SCU Through Several USB Ports on My PC?

No, this is not supported.

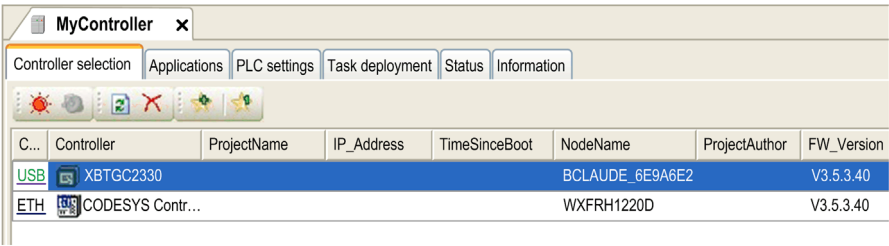
When I Use a New Controller in SoMachine Application with a Previously Used HMI Application, Why Do the 2 Applications No Longer Communicate?

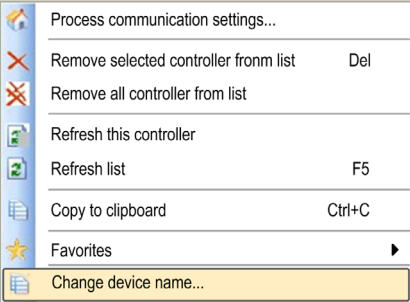
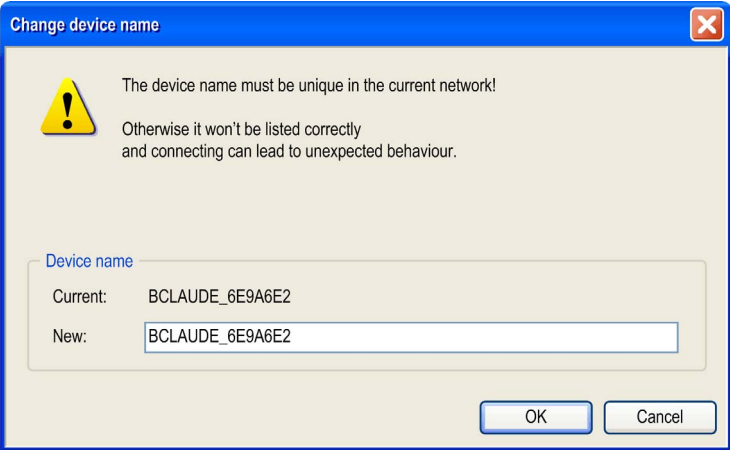
This is because the controller name in the HMI application (Vijeo-Designer) is not updated. The HMI application is configured with the previous controller name; it is necessary update this application with the SoMachine controller name.

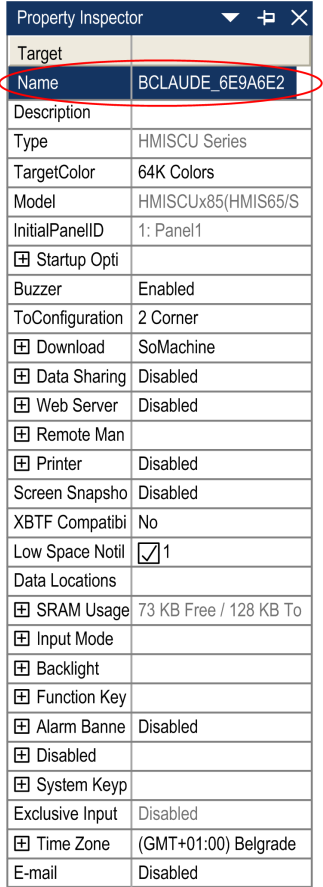
The following procedure updates the HMI application controller name with the SoMachine controller name. However, you may update the SoMachine controller name with the HMI application controller name, refer to update controller name using the HMI application (*see page 118*).

How Do I Manually Update my HMI Application Controller Name with the SoMachine Controller Name?

Copy the controller name from SoMachine application to the HMI Vijeo-Designer application controller name:

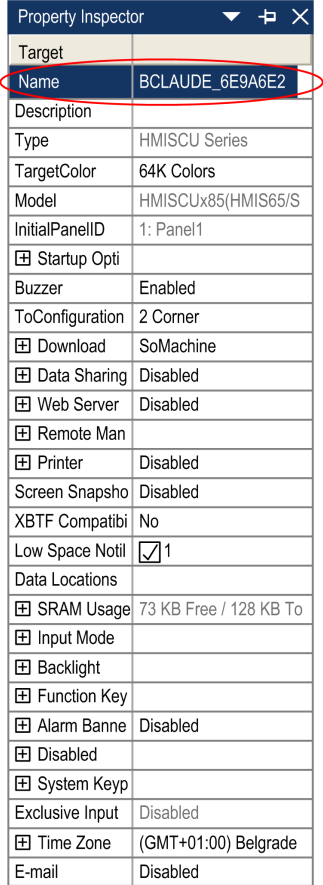
Step	Action																								
1	Display the SoMachine Logic Builder .																								
2	Double-click the controller in the Devices tree . Result: The device editor window opens.																								
3	Select the Controller selection tab. Result: The Controller selection tab opens:  <table border="1" data-bbox="336 1258 1227 1356"> <thead> <tr> <th>C...</th> <th>Controller</th> <th>ProjectName</th> <th>IP_Address</th> <th>TimeSinceBoot</th> <th>NodeName</th> <th>ProjectAuthor</th> <th>FW_Version</th> </tr> </thead> <tbody> <tr> <td>USB</td> <td>XBTGC2330</td> <td></td> <td></td> <td></td> <td>BCLAUDE_6E9A6E2</td> <td></td> <td>V3.5.3.40</td> </tr> <tr> <td>ETH</td> <td>CODESYS Contr...</td> <td></td> <td></td> <td></td> <td>WXFRH1220D</td> <td></td> <td>V3.5.3.40</td> </tr> </tbody> </table>	C...	Controller	ProjectName	IP_Address	TimeSinceBoot	NodeName	ProjectAuthor	FW_Version	USB	XBTGC2330				BCLAUDE_6E9A6E2		V3.5.3.40	ETH	CODESYS Contr...				WXFRH1220D		V3.5.3.40
C...	Controller	ProjectName	IP_Address	TimeSinceBoot	NodeName	ProjectAuthor	FW_Version																		
USB	XBTGC2330				BCLAUDE_6E9A6E2		V3.5.3.40																		
ETH	CODESYS Contr...				WXFRH1220D		V3.5.3.40																		

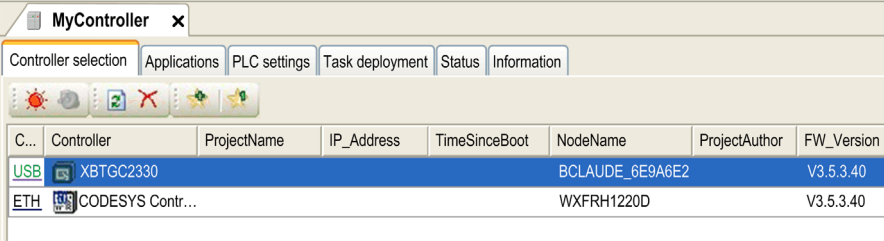
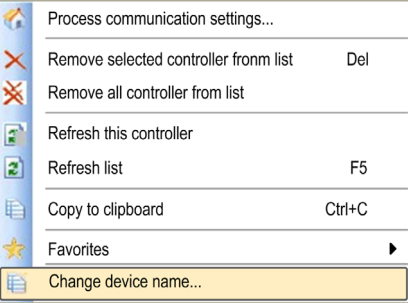
Step	Action
4	<p>Right-click the controller. Result: The controller contextual menu opens.</p> 
5	<p>Select <code>Change device name...</code> Result: The Change device name dialog opens:</p> 
6	<p>Make sure that device name meets the Vijeo-Designer controller name requirements: maximum length 32 characters (A-Z, a-z, 0-9, unicode characters, and _) and must start with a letter.</p>
7	<p>Copy the value contained in the New field.</p>
8	<p>Click OK.</p>
9	<p>Display the Vijeo-Frame.</p>


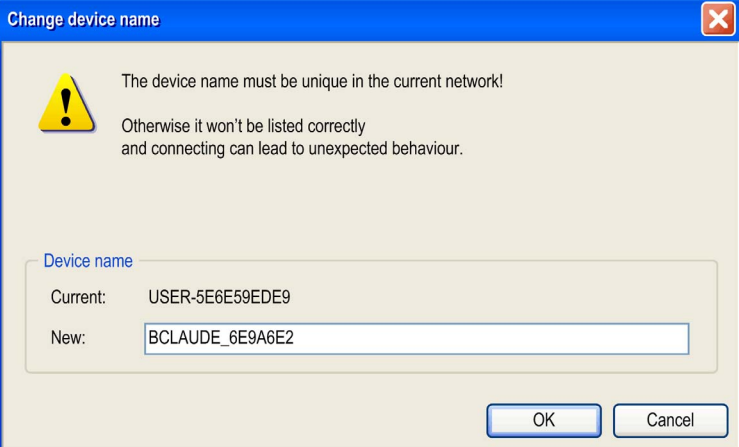
Step	Action
10	<p>Paste the Vijeo-Designer controller name in the Property Inspector → Name field:</p>  <p>The screenshot shows a 'Property Inspector' window with a list of properties. The 'Name' property is highlighted with a red oval and contains the text 'BCLAUDE_6E9A6E2'. Other properties include Description, Type (HMISCU Series), TargetColor (64K Colors), Model (HMISCUx85(HMIS65/S), InitialPanelID (1: Panel1), and various system settings like Buzzer (Enabled), ToConfiguration (2 Corner), and SRAM Usage (73 KB Free / 128 KB To).</p>
11	Press Enter to apply the change to the controller name.

How Do I Manually Update the SoMachine Controller Name with My HMI Application Controller Name?

Copy the controller name from the HMI Vijeo-Designer application to the SoMachine application controller name:

Step	Action																																																												
1	Display the Vijeo-Frame .																																																												
2	<p>Copy the Vijeo-Designer controller name from the Property Inspector → Name field:</p>  <table border="1"> <thead> <tr> <th colspan="2">Property Inspector</th> </tr> </thead> <tbody> <tr> <td>Target</td> <td></td> </tr> <tr> <td>Name</td> <td>BCLAUDE_6E9A6E2</td> </tr> <tr> <td>Description</td> <td></td> </tr> <tr> <td>Type</td> <td>HMISCU Series</td> </tr> <tr> <td>TargetColor</td> <td>64K Colors</td> </tr> <tr> <td>Model</td> <td>HMISCUx85(HMIS65/S</td> </tr> <tr> <td>InitialPanelID</td> <td>1: Panel1</td> </tr> <tr> <td>Startup Opti</td> <td></td> </tr> <tr> <td>Buzzer</td> <td>Enabled</td> </tr> <tr> <td>ToConfiguration</td> <td>2 Corner</td> </tr> <tr> <td>Download</td> <td>SoMachine</td> </tr> <tr> <td>Data Sharing</td> <td>Disabled</td> </tr> <tr> <td>Web Server</td> <td>Disabled</td> </tr> <tr> <td>Remote Man</td> <td></td> </tr> <tr> <td>Printer</td> <td>Disabled</td> </tr> <tr> <td>Screen Snapsho</td> <td>Disabled</td> </tr> <tr> <td>XBTF Compatibi</td> <td>No</td> </tr> <tr> <td>Low Space Notil</td> <td><input checked="" type="checkbox"/> 1</td> </tr> <tr> <td>Data Locations</td> <td></td> </tr> <tr> <td>SRAM Usage</td> <td>73 KB Free / 128 KB To</td> </tr> <tr> <td>Input Mode</td> <td></td> </tr> <tr> <td>Backlight</td> <td></td> </tr> <tr> <td>Function Key</td> <td></td> </tr> <tr> <td>Alarm Banne</td> <td>Disabled</td> </tr> <tr> <td>Disabled</td> <td></td> </tr> <tr> <td>System Keyp</td> <td></td> </tr> <tr> <td>Exclusive Input</td> <td>Disabled</td> </tr> <tr> <td>Time Zone</td> <td>(GMT+01:00) Belgrade</td> </tr> <tr> <td>E-mail</td> <td>Disabled</td> </tr> </tbody> </table>	Property Inspector		Target		Name	BCLAUDE_6E9A6E2	Description		Type	HMISCU Series	TargetColor	64K Colors	Model	HMISCUx85(HMIS65/S	InitialPanelID	1: Panel1	Startup Opti		Buzzer	Enabled	ToConfiguration	2 Corner	Download	SoMachine	Data Sharing	Disabled	Web Server	Disabled	Remote Man		Printer	Disabled	Screen Snapsho	Disabled	XBTF Compatibi	No	Low Space Notil	<input checked="" type="checkbox"/> 1	Data Locations		SRAM Usage	73 KB Free / 128 KB To	Input Mode		Backlight		Function Key		Alarm Banne	Disabled	Disabled		System Keyp		Exclusive Input	Disabled	Time Zone	(GMT+01:00) Belgrade	E-mail	Disabled
Property Inspector																																																													
Target																																																													
Name	BCLAUDE_6E9A6E2																																																												
Description																																																													
Type	HMISCU Series																																																												
TargetColor	64K Colors																																																												
Model	HMISCUx85(HMIS65/S																																																												
InitialPanelID	1: Panel1																																																												
Startup Opti																																																													
Buzzer	Enabled																																																												
ToConfiguration	2 Corner																																																												
Download	SoMachine																																																												
Data Sharing	Disabled																																																												
Web Server	Disabled																																																												
Remote Man																																																													
Printer	Disabled																																																												
Screen Snapsho	Disabled																																																												
XBTF Compatibi	No																																																												
Low Space Notil	<input checked="" type="checkbox"/> 1																																																												
Data Locations																																																													
SRAM Usage	73 KB Free / 128 KB To																																																												
Input Mode																																																													
Backlight																																																													
Function Key																																																													
Alarm Banne	Disabled																																																												
Disabled																																																													
System Keyp																																																													
Exclusive Input	Disabled																																																												
Time Zone	(GMT+01:00) Belgrade																																																												
E-mail	Disabled																																																												
3	Display the SoMachine Logic Builder .																																																												
4	<p>Double-click the controller in the Devices tree.</p> <p>Result: The device editor window opens.</p>																																																												

Step	Action																								
5	<p>Select the Controller selection tab.</p> <p>Result: The Controller selection tab opens:</p>  <table border="1" data-bbox="348 386 1226 477"> <thead> <tr> <th>C...</th> <th>Controller</th> <th>ProjectName</th> <th>IP_Address</th> <th>TimeSinceBoot</th> <th>NodeName</th> <th>ProjectAuthor</th> <th>FW_Version</th> </tr> </thead> <tbody> <tr> <td>USB</td> <td>XBTGC2330</td> <td></td> <td></td> <td></td> <td>BCLAUDE_6E9A6E2</td> <td></td> <td>V3.5.3.40</td> </tr> <tr> <td>ETH</td> <td>CODESYS Contr...</td> <td></td> <td></td> <td></td> <td>WXFRH1220D</td> <td></td> <td>V3.5.3.40</td> </tr> </tbody> </table>	C...	Controller	ProjectName	IP_Address	TimeSinceBoot	NodeName	ProjectAuthor	FW_Version	USB	XBTGC2330				BCLAUDE_6E9A6E2		V3.5.3.40	ETH	CODESYS Contr...				WXFRH1220D		V3.5.3.40
C...	Controller	ProjectName	IP_Address	TimeSinceBoot	NodeName	ProjectAuthor	FW_Version																		
USB	XBTGC2330				BCLAUDE_6E9A6E2		V3.5.3.40																		
ETH	CODESYS Contr...				WXFRH1220D		V3.5.3.40																		
6	<p>Right-click the controller.</p> <p>Result: The controller contextual menu opens.</p>  <table border="1" data-bbox="340 610 749 911"> <tbody> <tr> <td></td> <td>Process communication settings...</td> <td></td> </tr> <tr> <td></td> <td>Remove selected controller from list</td> <td>Del</td> </tr> <tr> <td></td> <td>Remove all controller from list</td> <td></td> </tr> <tr> <td></td> <td>Refresh this controller</td> <td></td> </tr> <tr> <td></td> <td>Refresh list</td> <td>F5</td> </tr> <tr> <td></td> <td>Copy to clipboard</td> <td>Ctrl+C</td> </tr> <tr> <td></td> <td>Favorites</td> <td>▶</td> </tr> <tr> <td></td> <td>Change device name...</td> <td></td> </tr> </tbody> </table>		Process communication settings...			Remove selected controller from list	Del		Remove all controller from list			Refresh this controller			Refresh list	F5		Copy to clipboard	Ctrl+C		Favorites	▶		Change device name...	
	Process communication settings...																								
	Remove selected controller from list	Del																							
	Remove all controller from list																								
	Refresh this controller																								
	Refresh list	F5																							
	Copy to clipboard	Ctrl+C																							
	Favorites	▶																							
	Change device name...																								

Step	Action
7	<p>Select Change device name...</p> <p>Result: The Change device name dialog opens:</p> 
8	<p>Paste the controller name into the New field.</p> 
9	<p>Click OK to apply the change to the controller name.</p>

How Do I Select the HMI SCU boot up Behavior (RUN or STOP) After a Power Cycle?

The RUN/STOP state of the HMI SCU depends on the status of the "Start all applications after download or online change" checkbox which appears when you use "Multiple download".

If it is checked, the HMI SCU boots to RUN. If not, it boots to STOP.

How Do I Create a Project Archive File

Create a project archive file by selecting **File** → **Project Archive** → **Save/Send Archive** from the SoMachine menu.

Why Does the Task Monitor Always Show Zero milliseconds for the Average and Minimum Task Times?

The HMI SCU only supports reporting back of cycle times to a 1 ms resolution, and requires a minimum of 2 ms for one HMI with a Control Process cycle. The CPU is scheduled to give HMI and Control each 1 ms (per 2 ms).

If a task requires less than 2 ms (2000 μ s) to run, the Task Monitor will show 0 μ s.

Appendices



Overview

This appendix lists the documents necessary for technical understanding of the HMI SCU Programming Guide.

What Is in This Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	Library <code>SE_ModbusTCP_Slave</code>	125
B	Function and Function Block Representation	131
C	Controller Performance	141

Appendix A

Library `SE_ModbusTCP_Slave`

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Presentation of the Library <code>SE_ModbusTCP_Slave</code>	126
Function block <code>ModbusServer</code>	127

Presentation of the Library SE_ModbusTCP_Slave

Overview

The library SE_ModbusTCP_Slave provides a communication function block, ModbusServer, for managing the communication between the Magelis SCU HMI Controller in the capacity of a Modbus server and any clients requesting Modbus services from the controller.

Compatibility

The library SE_ModbusTCP_Slave is compatible with the HMI SCU controllers HMISCUxA5 and HMISCUxB5 greater than or equal to firmware version 3.5.3.50.

The SE_ModbusTCP_Slave library has not been validated on any other SoMachine controller platform.

WARNING

UNINTENDED EQUIPMENT OPERATION

Do not add the SE_ModbusTCP_Slave library to programs other than those that target the HMISCUxA5 or HMISCUxB5 controllers.

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

Installation of the Library

If the library is not present in your SoMachine application, add the library SE_ModbusTCP_Slave via the library manager.

For more information on the library manager, refer to Add Library (*see SoMachine, Functions and Libraries User Guide*).

Function block ModbusServer

Presentation

The function block `ModbusServer` is part of the library `SE_ModbusTCP_Slave`. It handles ModbusTCP requests for writing and reading data between the HMI SCU controller as a server and Modbus clients. Up to 10 client connections can be served. The function block operates on defined data areas in the HMI SCU that represent Modbus input and output registers.

The `ModbusServer` function block runs in a POU assigned to a task, and therefore, if the controller is in STOP mode, the task with the `ModbusServer` function block is not running.

The Unit ID of the `ModbusServer` is 255.

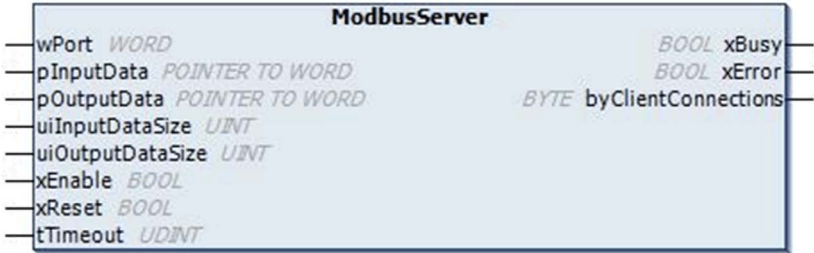
Supported Function Codes

The following Modbus function codes are supported:

Function code	Description
3 (03 hex) or 4 (04 hex)	Read Multiple Registers
6 (06 hex)	Write Single Register
16 (10 hex)	Write Multiple Registers
23 (17 hex)	Read/Write Multiple Registers

Up to 10 client connections can be managed together. All client requests are processed within one controller cycle; however, one request per client connection.

Graphical Representation



Parameters

This table describes the inputs of the function block:

Input	Type	Default Value	Description
wPort	WORD	502	Server port, 502 is Modbus default ¹
pInputData	POINTER TO WORD	-	First word address of the input data table.
pOutputData	POINTER TO WORD	-	First word address of the output data table.
uiInputDataSize	UINT	-	Size of the input data table.
uiOutputDataSize	UINT	-	Size of the output data table.
xEnable	BOOL	FALSE	When TRUE , the function block is executed. The values of the function block inputs can be modified and its outputs are updated continuously. When FALSE , terminates the function block execution and resets its outputs.
xReset	BOOL	FALSE	When TRUE , re-enables communication and the error bit.
tTimeout	UDINT	0	Modbus write timeout in ms. The function block needs to be reset after a timeout to re-enable communication.
<p>1 Port 502 is the default port used for Modbus TCP/IP protocol. If Modbus TCP/IP (server or client) protocol is already configured in Vijeo-Designer for the controller, use a different port in the function block, to communicate between the server and the clients.</p>			

This table describes the outputs of the function block:

Output	Type	Default Value	Description
xBusy	BOOL	-	When TRUE , the function block is servicing a client request. When FALSE , the function block has terminated servicing a client request. The function block must be kept in an active task of the application program for at least as long as xBusy is TRUE . If not, the data tables are set to 0 while the function block outputs remain static.
xError	BOOL	-	When TRUE , it indicates that an error was detected. The execution of the function block has been terminated. The function block needs to be reset to re-enable communication.
byClientConnections	BYTE	0	Number of active connections.

Return Codes

This table lists the values for the return codes for the library SE_ModbusTCP_Slave which are sent to the clients for various erroneous requests, or in response to successful operations:

Name	Type	Value	Description
RESPONSE_SUCCESS	BYTE	16#0	Successful communication.
ILLEGAL_FUNCTION	BYTE	16#1	The function code is not supported by the server.
ILLEGAL_DATA_ADDRESS	BYTE	16#2	The register offset is not supported by the server.
ILLEGAL_DATA_VALUE	BYTE	16#3	Incorrect value.
SLAVE_DEVICE_FAILURE	BYTE	16#4	An error has been detected on the server device.
ACKNOWLEDGE	BYTE	16#5	Server response to valid client request.
SLAVE_DEVICE_BUSY	BYTE	16#6	Server response to client request sent while an operation is already in progress.
GATEWAY_PATH_UNAVAILABLE	BYTE	16#A	Error detected when communicating through a gateway (Gateway misconfigured/busy)
GATEWAY_DEVICE_FAILED_TO_RESPOND	BYTE	16#B	Device connected to gateway did not respond.

Global Constant List

This table present the Global Constant List (GCL) for the library SE_ModbusTCP_Slave:

Name	Type	Value	Description
MODBUS_TCP_MAXCLIENTS	UINT	10	Non-editable GCL containing the maximum number of active connections.

Appendix B

Function and Function Block Representation

Overview

Each function can be represented in the following languages:

- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram
- CFC: Continuous Function Chart

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Differences Between a Function and a Function Block	132
How to Use a Function or a Function Block in IL Language	133
How to Use a Function or a Function Block in ST Language	137

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (AND), calculations, conversion (BYTE_TO_INT)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, `Timer_ON` is an instance of the function block `TON`:

```
1  PROGRAM MyProgram_ST
2  VAR
3      Timer_ON: TON; // Function Block Instance
4      Timer_RunCd: BOOL;
5      Timer_PresetValue: TIME := T#5S;
6      Timer_Output: BOOL;
7      Timer_ElapsedTime: TIME;
8  END_VAR
```

```
1  Timer_ON(
2      IN:=Timer_RunCd,
3      PT:=Timer_PresetValue,
4      Q=>Timer_Output,
5      ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. NOTE: The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POU's (<i>see SoMachine, Programming Guide</i>).
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: <ul style="list-style-type: none"> type the name of the function in the operator column (left field), or use the Input Assistant to select the function (select Insert Box in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with ??? in the fields on the right. Replace the ??? with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: <code>IsFirstMastCycle</code>	
with input parameters: <code>SetRTCDrift</code>	

In IL language, the function name is used directly in the operator column:

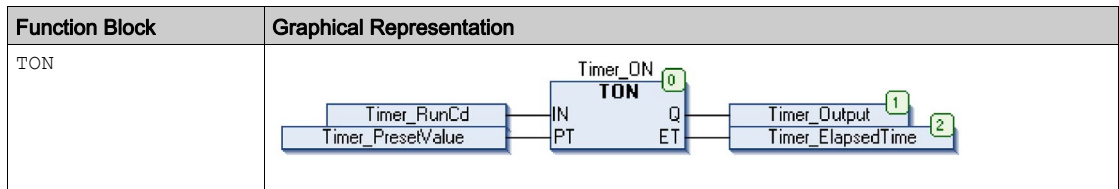
Function	Representation in SoMachineSoMachine BasicSoMachine Motion POU IL Editor															
IL example of a function without input parameter: IsFirstMastCycle	<pre> 1 PROGRAM MyProgram_IL 2 VAR 3 FirstCycle: BOOL; 4 END_VAR </pre> <hr/> <table border="1" data-bbox="371 456 979 570"> <tr> <td data-bbox="371 456 444 493">1</td> <td data-bbox="444 456 742 493">IsFirstMast Cycle</td> <td data-bbox="742 456 979 493"></td> </tr> <tr> <td data-bbox="371 493 444 531"></td> <td data-bbox="444 493 742 531">ST</td> <td data-bbox="742 493 979 531">FirstCycle</td> </tr> <tr> <td data-bbox="371 531 444 568"></td> <td data-bbox="444 531 742 568"></td> <td data-bbox="742 531 979 568"></td> </tr> </table>	1	IsFirstMast Cycle			ST	FirstCycle									
1	IsFirstMast Cycle															
	ST	FirstCycle														
IL example of a function with input parameters: SetRTCDrift	<pre> 1 PROGRAM MyProgram_IL 2 VAR 3 myDrift: SINT (-29..29) := 5; 4 myDay: DAY_OF_WEEK := SUNDAY; 5 myHour: HOUR := 12; 6 myMinute: MINUTE; 7 myDiag: RTCSETDRIFT_ERROR; 8 END_VAR </pre> <hr/> <table border="1" data-bbox="371 967 930 1146"> <tr> <td data-bbox="371 967 444 1005">1</td> <td data-bbox="444 967 687 1005">LD</td> <td data-bbox="687 967 930 1005">myDrift</td> </tr> <tr> <td data-bbox="371 1005 444 1042"></td> <td data-bbox="444 1005 687 1042">SetRTCDrift</td> <td data-bbox="687 1005 930 1042">myDay</td> </tr> <tr> <td data-bbox="371 1042 444 1079"></td> <td data-bbox="444 1042 687 1079"></td> <td data-bbox="687 1042 930 1079">myHour</td> </tr> <tr> <td data-bbox="371 1079 444 1117"></td> <td data-bbox="444 1079 687 1117"></td> <td data-bbox="687 1079 930 1117">myMinute</td> </tr> <tr> <td data-bbox="371 1117 444 1154"></td> <td data-bbox="444 1117 687 1154">ST</td> <td data-bbox="687 1117 930 1154">myDiag</td> </tr> </table>	1	LD	myDrift		SetRTCDrift	myDay			myHour			myMinute		ST	myDiag
1	LD	myDrift														
	SetRTCDrift	myDay														
		myHour														
		myMinute														
	ST	myDiag														

Using a Function Block in IL Language

This procedure describes how to insert a function block in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. NOTE: The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs (<i>see SoMachine, Programming Guide</i>).
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a CAL instruction: <ul style="list-style-type: none"> ● Use the Input Assistant to select the FB (right-click and select Insert Box in the context menu). ● Automatically, the CAL instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: <ul style="list-style-type: none"> ● Values to inputs are set by " := ". ● Values to outputs are set by " => ".
4	In the CAL right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the **TON** Function Block graphically presented below:



In IL language, the function block name is used directly in the operator column:

Function Block	Representation in SoMachineSoMachine BasicSoMachine Motion POU IL Editor
TON	<pre> 1 PROGRAM MyProgram_IL 2 VAR 3 Timer_ON: TON; // Function Block instance declaration 4 Timer_RunCd: BOOL; 5 Timer_PresetValue: TIME := T#5S; 6 Timer_Output: BOOL; 7 Timer_ElapsedTime: TIME; 8 END_VAR 9 </pre> <hr/> <pre> 1 CAL Timer_ON(IN:= Timer_RunCd, PT:= Timer_PresetValue, Q=> Timer_Output, ET=> Timer_ElapsedTime) </pre>

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

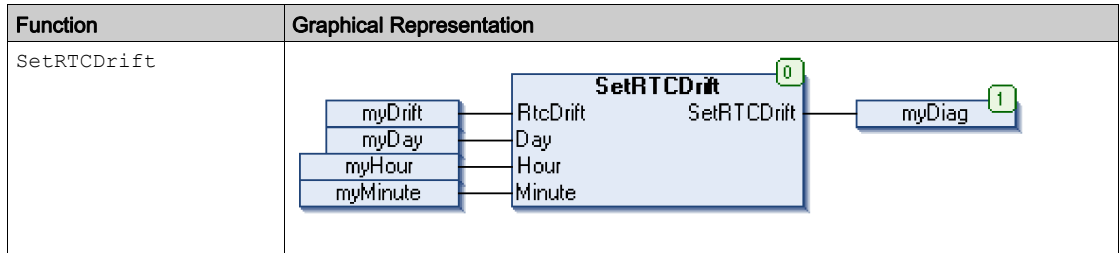
Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. NOTE: The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POU's (<i>see SoMachine, Programming Guide</i>).
2	Create the variables that the function requires.
3	Use the general syntax in the POU ST Editor for the ST language of a function. The general syntax is: FunctionResult:= FunctionName (VarInput1, VarInput2,.. VarInputx);

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:



The ST language of this function is the following:

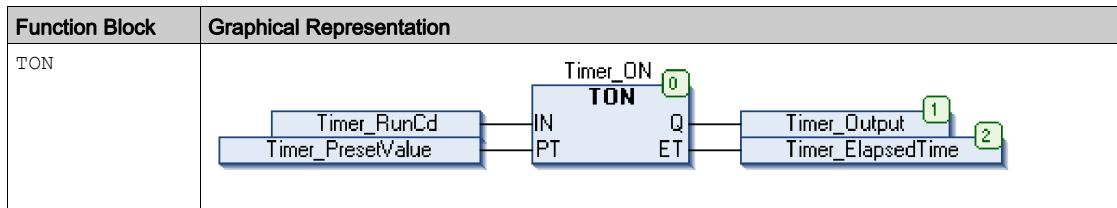
Function	Representation in SoMachineSoMachine BasicSoMachine Motion POU ST Editor
SetRTCDrift	<pre>PROGRAM MyProgram_ST VAR myDrift: SINT(-29..29) := 5; myDay: DAY_OF_WEEK := SUNDAY; myHour: HOUR := 12; myMinute: MINUTE; myRTCAdjust: RTCDRIFT_ERROR; END_VAR myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);</pre>

Using a Function Block in ST Language

This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. NOTE: The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation (<i>see SoMachine, Programming Guide</i>).
2	Create the input and output variables and the instance required for the function block: <ul style="list-style-type: none"> • Input variables are the input parameters required by the function block • Output variables receive the value returned by the function block
3	Use the general syntax in the POU ST Editor for the ST language of a Function Block. The general syntax is: <pre>FunctionBlock_InstanceName (Input1:=VarInput1, Input2:=VarInput2, ... Ouput1=>VarOutput1, Ouput2=>VarOutput2, ...);</pre>

To illustrate the procedure, consider this example with the TON function block graphically presented below:



This table shows examples of a function block call in ST language:

Function Block	Representation in SoMachineSoMachine BasicSoMachine Motion POU ST Editor
TON	<pre> 1 PROGRAM MyProgram_ST 2 VAR 3 Timer_ON: TON; // Function Block Instance 4 Timer_RunCd: BOOL; 5 Timer_PresetValue: TIME := T#5S; 6 Timer_Output: BOOL; 7 Timer_ElapsedTime: TIME; 8 END_VAR 1 Timer_ON(2 IN:=Timer_RunCd, 3 PT:=Timer_PresetValue, 4 Q=>Timer_Output, 5 ET=>Timer_ElapsedTime); </pre>

Appendix C

Controller Performance

Processing Performance

Introduction

This chapter provides information about the HMI SCU controller processing performance.

Logic Processing

The table shows logic processing performance for various instruction type:

Instruction Type	Average duration for 1000 instructions with 1000 cycles
Addition/subtraction/multiplication of INT (IL)	30 μ s
Addition/subtraction/multiplication of DINT (IL)	20 μ s
Addition/subtraction/multiplication of REAL (IL)	600 μ s
Division of INT (IL)	791 μ s
Division of DINT (IL)	790 μ s
Division of REAL (IL)	1400 μ s
Operation on BOOLEAN (Bool1=Bool2 & Bool3) (ST)	56 μ s
LD + ST INT (IL)	49 μ s
LD + ST DINT (IL)	10 μ s
LD + ST REAL (IL)	20 μ s

Basic System Time

The table shows the basic overhead performance for each MAST cycle:

I/O Type	Overhead for each MAST cycle
Embedded Inputs & Internal processing	1500 μ s
Embedded outputs	450 μ s

HSC, PWM, and PTO Processing

The table shows the processing performance for complex functions for each MAST cycle:

Complex Function Type	Overhead for each MAST cycle
HSC Simple	150 μ s
HSC Main	350 μ s
PWM	150 μ s
PTO Simple	200 μ s

Communication and System Processing Time

The communication processing time varies, depending on the number of sent/received requests.

Response Time on Event

The response time shown in the following table represents the time between a signal rising edge on an input triggering an external task and the edge of an output set by this task. The event task also process 100 IL instructions before setting the output:

Minimum	Typical	Maximum
800 μ s	4000 μ s	4300 μ s



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.

ASCII

(American standard code for Information Interchange) A protocol for representing alphanumeric characters (letters, numbers, certain graphics, and control characters).

B

BCD

(binary coded decimal) The format that represents decimal numbers between 0 and 9 with a set of 4 bits (a nybble/nibble, also titled as half byte). In this format, the 4 bits used to encode decimal numbers have an unused range of combinations.

For example, the number 2,450 is encoded as 0010 0100 0101 0000.

BOOL

(boolean) A basic data type in computing. A `BOOL` variable can have one of these values: 0 (`FALSE`), 1 (`TRUE`). A bit that is extracted from a word is of type `BOOL`; for example, `%MW10.4` is a fifth bit of memory word number 10.

Boot application

(boot application) The binary file that contains the application. Usually, it is stored in the controller and allows the controller to boot on the application that the user has generated.

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

CAN

(*controller area network*) A protocol (ISO 11898) for serial bus networks, designed for the interconnection of smart devices (from multiple manufacturers) in smart systems and for real-time industrial applications. Originally developed for use in automobiles, CAN is now used in a variety of industrial automation control environments.

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

configuration

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

control network

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

Two kinds of topologies are supported:

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

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

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.)

DINT

(*double integer type*) Encoded in 32-bit format.

DTM

(*device type manager*) Classified into 2 categories:

- Device DTMs connect to the field device configuration components.
- CommDTMs connect to the software communication components.

The DTM provides a unified structure for accessing device parameters and configuring, operating, and diagnosing the devices. DTMs can range from a simple graphical user interface for setting device parameters to a highly sophisticated application capable of performing complex real-time calculations for diagnosis and maintenance purposes.

DWORD

(*double word*) Encoded in 32-bit format.

E**equipment**

A part of a machine including sub-assemblies such as conveyors, turntables, and so on.

Ethernet

A physical and data link layer technology for LANs, also known as IEEE 802.3.

expansion bus

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

F**FB**

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

FBD

(*function block diagram*) One of 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks, where each network contains a graphical structure of boxes and connection lines, which represents either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

firmware

Represents the BIOS, data parameters, and programming instructions that constitute the operating system on a controller. The firmware is stored in non-volatile memory within the controller.

freewheeling

When a logic controller is in freewheeling scan mode, a new task scan starts as soon as the previous scan has been completed. Contrast with *periodic scan mode*.

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

H

HSC

(*high-speed counter*)

I

I/O

(*input/output*)

IEC

(*international electrotechnical commission*) A non-profit and non-governmental international standards organization that prepares and publishes international standards for electrical, electronic, and related technologies.

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).

INT

(*integer*) A whole number encoded in 16 bits.

IP

(*Internet protocol*) Part of the TCP/IP protocol family that tracks the Internet addresses of devices, routes outgoing messages, and recognizes incoming messages.

L

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).

LINT

(long integer) A whole number encoded in a 64-bit format (4 times `INT` or 2 times `DINT`).

LREAL

(long real) A floating-point number encoded in a 64-bit format.

LWORD

(long word) A data type encoded in a 64-bit format.

M**MAC address**

(media access control address) A unique 48-bit number associated with a specific piece of hardware. The MAC address is programmed into each network card or device when it is manufactured.

MAST

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

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

Modbus

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

ms

(millisecond)

N**network**

A system of interconnected devices that share a common data path and protocol for communications.

NMT

(network management) CANopen protocols that provide services for network initialization, detected error control, and device status control.

P**PDO**

(process data object) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

POU

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

program

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

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. The 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). The PTO is well adapted to simulate or approximate an analog output in that it regulates the voltage of the output over its period making it useful in light dimming or speed control applications, among others.

R

REAL

A data type that is defined as a floating-point number encoded in a 32-bit format.

RPDO

(receive process data object) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

S

SDO

(service data object) A message used by the field bus master to access (read/write) the object directories of network nodes in CAN-based networks. SDO types include service SDOs (SSDOs) and client SDOs (CSDOs).

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.)

SINT

(signed integer) A 15-bit value plus sign.

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

STOP

A command that causes the controller to stop running an application program.

string

A variable that is a series of ASCII characters.

T**task**

A group of sections and subroutines, executed cyclically or periodically for the MAST task or periodically for the FAST task.

A task possesses a level of priority and is linked to inputs and outputs of the controller. These I/O are refreshed in relation to the task.

A controller can have several tasks.

TPDO

(*transmit process data object*) An unconfirmed broadcast message or sent from a producer device to a consumer device in a CAN-based network. The transmit PDO from the producer device has a specific identifier that corresponds to the receive PDO of the consumer devices.

U**UDINT**

(*unsigned double integer*) Encoded in 32 bits.

UINT

(*unsigned integer*) Encoded in 16 bits.

V**variable**

A memory unit that is addressed and modified by a program.

W**WORD**

A type encoded in a 16-bit format.



A

- analog I/O
 - analog I/O embedded function, *77*
 - analog temperature embedded function, *82*
- Application
 - Save, *104*

C

- Configuration
 - IP Address Configuration, *88*
- Controller
 - Connecting the controller, *103*
- Controller Configuration
 - Applications, *64*
 - Controller Selection, *63*
 - PLC Settings, *65*

D

- Download
 - Application, *103*
- Download application, *56*

E

- Embedded Functions Configuration
 - Embedded HSC Configuration, *72*
 - Embedded I/O Configuration, *68*
- embedded functions configuration
 - embedded PTO_PWM configuration, *74*
- Ethernet
 - Modbus TCP Client/Server, *90*
- External Event, *34*

F

- FAQ, *113*
 - Connecting Multiple Controller through

- USB Ports, *115*
 - Controller and HMI Communication, *115*
 - Controller Boot State, *121*
 - Controller Name Update, *115, 118*
 - SoMachine Network Communication, *113*
 - Start all application checkboxes, *115*
 - Supported Programming Languages, *113*
 - Supported Variables, *113*
 - Task Mode, *113*
 - Task Monitor, *121*
 - Watchdog Configuration, *114*

features

- key features, *11*

Firmware

- Downgrade, *103*
- Save, *104*
- Update, *103*

functions

- differences between a function and a function block, *132*
- how to use a function or a function block in IL language, *133*
- how to use a function or a function block in ST language, *137*

H

- Hardware Initialization Values, *51*

I

- IP Address
 - Configuration, *88*
 - Default, *89*

L

- libraries, *19*

M

- main features, *11*
- Memory Mapping, *23*
- Modbus
 - Protocols, *90*
- Modbus TCP Client/Server
 - Ethernet, *90*

O

- Output Behavior, *51, 51, 52*
- Output Forcing, *52*
- overview, *11*

P

- programming languages
 - IL, ST, FBD, SFC, LD, CFC, *11*
- Protocols
 - Modbus, *90*

R

- Reboot, *55*
 - Transfer, *103*
- Remanent variables, *59*
- Reset cold, *54*
- Reset origin, *55*
- Reset warm, *54*
- Run command, *53*

S

- Save
 - Application, *104*
 - Firmware, *104*
 - USB, *104*
- Serial Line
 - Serial Line Configuration, *92*
- Software Initialization Values, *51*
- State diagram, *42*
- Stop command, *53*

T

- Task
 - Cyclic task, *32*
 - Event task, *34*
 - External Event Task, *34*
 - Freewheeling task, *33*
 - Types, *32*
 - Watchdogs, *36*
- Troubleshooting, *108*
 - Application Transfer, *108*
 - Boot Application, *111*
 - CANopen Heartbeat, *111*
 - Communication, *108*
 - Device Name, *111*
 - Out of Memory, *112*
 - POU Monitoring, *112*
 - RUN State, *111*

U

- USB
 - Save, *104*