# Magelis
# XBTGC HMI Controller
## Programming Guide

12/2016

EIO0000000632.08

www.schneider-electric.com

Schneider Electric

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

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

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

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

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

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

© 2016 Schneider Electric. All Rights Reserved.

# Table of Contents


| | | |
|---|---|---|
| | **Safety Information** | 5 |
| | **About the Book** | 7 |
| **Chapter 1** | **Starting with a New Project** | 11 |
| 1.1 | New Project | 12 |
| | Creating a New Project | 13 |
| | Trees Description | 15 |
| 1.2 | Adding Devices to the Project | 16 |
| | Adding an XBTGC HMI Controller | 17 |
| | Adding a CANopen Expansion Module | 18 |
| | Adding Expansion Modules | 19 |
| **Chapter 2** | **Libraries** | 21 |
| | Libraries | 21 |
| **Chapter 3** | **Supported Standard Data Types** | 23 |
| | Supported Variables | 24 |
| | Variables Exchange | 26 |
| **Chapter 4** | **Controller Memory Mapping** | 27 |
| | Memory Mapping | 28 |
| | Controllers and HMI Address Mapping Differences | 29 |
| **Chapter 5** | **Tasks** | 31 |
| | Maximum Number of Tasks | 32 |
| | Task Configuration Screen | 33 |
| | Task Types | 36 |
| | System and Task Watchdogs | 38 |
| | Task Priorities | 39 |
| | Default Task Configuration | 41 |
| **Chapter 6** | **Controller States and Behaviors** | 43 |
| 6.1 | Controller State Diagram | 44 |
| | Controller State Diagram | 44 |
| 6.2 | Controller States Description | 48 |
| | Controller States Description | 48 |
| 6.3 | State Transitions and System Events | 51 |
| | Controller States and Output Behavior | 52 |
| | Commanding State Transitions | 55 |
| | Error Detection, Types, and Management | 60 |
| | Remanent Variables | 62 |

Chapter 7 Controller Configuration . . . . . 63
Device Editor . . . . . 63
Chapter 8 Embedded I/O Configuration . . . . . 65
Embedded I/O Configuration Editor . . . . . 65
Chapter 9 Special I/O Configuration . . . . . 69
Local and Special I/O Overview . . . . . 70
Special I/O Configuration Possibilities . . . . . 72
I/O Summary . . . . . 76
Chapter 10 Expansion Modules Configuration . . . . . 79
10.1 I/O Configuration . . . . . 80
General Considerations . . . . . 80
10.2 Digital I/O Modules . . . . . 81
TM2 Digital I/O Modules . . . . . 81
10.3 Analog I/O Modules . . . . . 82
TM2 Analog I/O Modules . . . . . 82
Chapter 11 Ethernet Configuration . . . . . 83
IP Address Configuration . . . . . 83
Chapter 12 CANopen Configuration . . . . . 85
CANopen Interface Configuration . . . . . 86
CANopen Optimized Manager . . . . . 88
CANopen Remote Devices . . . . . 89
Chapter 13 Serial Line Configuration . . . . . 91
Serial Line Configuration . . . . . 92
SoMachine Network Manager . . . . . 94
Modbus Manager . . . . . 95
Chapter 14 Managing Online Applications . . . . . 97
Connecting the Controller to a PC . . . . . 97
Chapter 15 Troubleshooting and FAQ . . . . . 103
Troubleshooting . . . . . 104
Frequently Asked Questions . . . . . 108
Glossary . . . . . 117
Index . . . . . 127

# 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:

- show you how to program and operate your XBTGC HMI Controller,
- help you to understand how to program your XBTGC HMI Controller functions,
- help you to become familiar with the XBTGC HMI Controller functions.

Read and understand this document and all related documents before installing, operating or maintaining your XBTGC HMI Controller.

### Validity Note

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

### Related Documents

| Title of Documentation | Reference Number |
|---|---|
| SoMachine Programming Guide | EIO0000000067 (ENG);<br>EIO0000000069 (FRE);<br>EIO0000000068 (GER);<br>EIO0000000071 (SPA);<br>EIO0000000070 (ITA);<br>EIO0000000072 (CHS) |
| Magelis XBTGC HMI Controller Hardware Guide | 35016393 (ENG);<br>35016400 (FRE);<br>35016401 (GER);<br>35016402 (SPA);<br>35016403 (ITA);<br>35016404 (CHS) |
| Modicon TM2 Expansion Modules Configuration Programming Guide | EIO0000000396 (ENG);<br>EIO0000000397 (FRE);<br>EIO0000000398 (GER);<br>EIO0000000399 (SPA);<br>EIO0000000400 (ITA);<br>EIO0000000401 (CHS) |

| Title of Documentation | Reference Number |
|---|---|
| Magelis XBT Gx HMI Controller System Functions and Variables XBT PLCSystem Library Guide | EIO0000000626 (ENG);<br>EIO0000000627 (FRE);<br>EIO0000000628 (GER);<br>EIO0000000629 (SPA);<br>EIO0000000630 (ITA);<br>EIO0000000631 (CHS) |
| Magelis XBTGC HMI Controller High Speed Counting XBTGC HSC Library Guide | EIO0000000644 (ENG);<br>EIO0000000645 (FRE);<br>EIO0000000646 (GER);<br>EIO0000000647 (SPA);<br>EIO0000000648 (ITA);<br>EIO0000000649 (CHS) |
| Magelis XBTGC HMI Controller Pulse Train Output, Pulse Width Modulation XBTGC PTOPWM Library Guide | EIO0000000650 (ENG);<br>EIO0000000651 (FRE);<br>EIO0000000652 (GER);<br>EIO0000000653 (SPA);<br>EIO0000000654 (ITA);<br>EIO0000000655 (CHS) |
| SoMachine Modbus and ASCII Read/Write Functions PLCCommunication Library Guide | EIO0000000361 (ENG);<br>EIO0000000742 (FRE);<br>EIO0000000743 (GER);<br>EIO0000000744 (SPA);<br>EIO0000000745 (ITA);<br>EIO0000000746 (CHS) |

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.**

# Chapter 1
# Starting with a New Project

## Introduction

This chapter describes how to create a project with the XBTGC HMI Controller and how to add devices.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 1.1 | New Project | 12 |
| 1.2 | Adding Devices to the Project | 16 |

# Section 1.1
# New Project

## Introduction

This section will guide you through creating a new XBTGC HMI Controller project.

## What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Creating a New Project | 13 |
| Trees Description | 15 |

# Creating a New Project

## Introduction

This section describes the general characteristics of the XBTGC HMI Controller and how to create a new SoMachine project. Refer to *Manage your project (see SoMachine Central, User Guide)* for additional information.

The XBTGC HMI Controller integrates both HMI interface (configured using Vijeo-Designer) and controller features (configured using SoMachine).

## XBTGC HMI Controller Main Characteristics

This table lists the main characteristics of the XBTGC HMI Controller:

| | XBTGC1100 | XBTGC2120 | XBTGC2230/XBTGC2330 |
|---|---|---|---|
| **Embedded inputs** | 12 | 16 | 16 |
| **Embedded outputs** | 6 | 16 | 16 |
| **Display type** | Monochrome Amber/Red LCD | Monochrome LCD | STN/TFT Color LCD |
| **Expansion modules** | 2 max. | 3 max. | 3 max. |
| **Ethernet interface** | Not available | Not available | Available |
| **Serial interface (COM1)** | Not available | RS232/RS422/RS485 serial interface. SUB-D 9-pin plug connector. | RS232/RS422/RS485 serial interface. SUB-D 9-pin plug connector. |
| **USB Interface** | Available | Available | Available |

**NOTE:** Refer to *Controller Specifications (see Magelis XBTGC HMI Controller, Hardware Guide)* for additional information on the controller hardware.

## Creating a New Project

To create a new project, you must add a controller to the **Devices tree**. Refer to *Devices Tree Description (see page 15)* and to *Adding a Controller (see page 17)*.

## Active Application

The active application is displayed in bold print in the **Applications tree**. When working on a project that contains several applications, check that the application you are currently working on is activated. Certain commands (for example, the **Build** command) are by default executed on the active application.

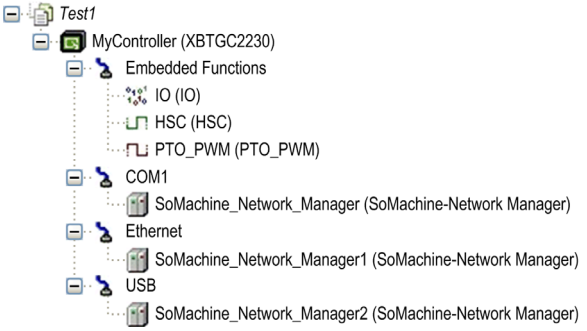
To activate an application, right-click its entry in the **Applications tree** and select **Set Active Application** from the context menu.

**NOTE:** Using **Set Active Application** during multiple application controls (not HMI applications) changes the description of several commands in the **Build** menu, in order to refer to the new active application.

# Trees Description

## 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.




This table describes the items in the **Devices tree**.

| Item | Description |
|---|---|
| Embedded Functions | Embedded functions include:<br>• **IO:** Configuration of the embedded I/O<br>• **HSC:** Configuration of the High Speed Counter<br>• **PTO_PWM:** Configuration of the Pulse Train Output and Pulse Width Modulation |
| COM1 | Embedded communication functions for Serial Line *(see page 91)* communication. |
| Ethernet | Embedded communication functions for Ethernet *(see page 83)* communication. |
| USB | Embedded communication functions for USB communication. |

## 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.

# Section 1.2
## Adding Devices to the Project

### Introduction

This section shows you how to add devices to your project.

### What Is in This Section?

This section contains the following topics:


| Topic | Page |
|---|---|
| Adding an XBTGC HMI Controller | 17 |
| Adding a CANopen Expansion Module | 18 |
| Adding Expansion Modules | 19 |

# Adding an XBTGC HMI Controller

## Introduction

The following paragraphs explain how to add the XBTGC HMI Controller to a SoMachine project.

## Adding the XBTGC HMI Controller to the Devices Tree

To add an XBTGC HMI Controller to your project, select an **XBTGC••••** controller in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*
- Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

# Adding a CANopen Expansion Module

## Introduction

You can add a XBTZGCCAN CANopen expansion module with the XBTGC HMI Controller.

The CANbus node is automatically created. You can then add and configure further CANopen devices to the manager.

Adding a CANopen expansion is explained in CANopen Interface Configuration *(see page 86)*.

# Adding Expansion Modules

## Introduction

The following paragraph shows you how to add analog or digital I/O expansion modules to the XBTGC HMI Controller.

**⚠ 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.**

## XBTGC HMI Controller Maximum Hardware Configuration

The total width of all expansion modules attached to the controller must not exceed 60 mm (2.36 in) to maintain an acceptable level of vibration and shock resistance.

***NOTICE***

**EQUIPMENT DISCONNECTION**

Ensure that the total width of the expansion modules does not exceed 60 mm (2.36 in).

**Failure to follow these instructions can result in equipment damage.**

The *number of allowed modules (see Magelis XBTGC HMI Controller, Hardware Guide)* is reduced when adding large-size modules.

**NOTE:** In the hardware configuration, it is not physically possible to have a set of I/O expansion modules and a CANopen module together mounted on the back of the XBTGC HMI Controller.

## Adding Expansion Module to The XBTGC HMI Controller

To add an expansion module to your controller, select the expansion module in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

• Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

# Chapter 2
# Libraries

## Libraries

### Introduction

The libraries of the controller provide functions such as function blocks, data types and global variables that can be used to develop your project. The default extension for a library is ".library".

The **Library Manager** of SoMachine provides information about the libraries included in your project. You can also use the **Library Manager** to install new libraries.

For more information on the **Library Manager**, refer to the SoMachine Programming Guide.

### XBTGC HMI Controller Libraries

When you select an XBTGC HMI Controller for your application, SoMachine automatically loads the following libraries:

- **IoStandard:CmpIoMgr** configures types, access, parameters and help functions
- **Standard:** Bistable function blocks, counter, miscellaneous, string functions, timer and trigger
- **Util:** Analog monitors, BCD Conversions, Bit/Byte functions, controller datatypes, function manipulators, mathematical functions and signals
- **PLCCommunication:** Enables communication and it is common to all controller
- **XBT PLCSystem:** Refer to *XBT PLCSystem Library*
- **XBTGC HSC:** Refer to *XBTGC HSC Library*
- **XBTGC PTOPWM:** Refer to *XBTGC PTO/PWM Library*

# Chapter 3
# Supported Standard Data Types

## Introduction

This chapter provides the supported variables and explains how to exchange data between SoMachine (controller part) and Vijeo-Designer (HMI part).

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Supported Variables | 24 |
| Variables Exchange | 26 |

# Supported Variables

## Supported Variables Types

This table provides the XBTGC HMI Controller supported variables types:

| Controller Data Type | Lower Limit | Upper Limit | Information Content | Bidirectional Variable (SoMachine/Vijeo-Designer) |
|---|---|---|---|---|
| BOOL | False | True | 1 Bit | Yes |
| BYTE | 0 | 255 | 8 Bit | Yes |
| WORD | 0 | 65,535 | 16 Bit | Yes |
| DWORD | 0 | 4,294,967,295 | 32 Bit | Yes |
| LWORD | 0 | $2^{64}$-1 | 64 Bit | No |
| SINT | -128 | 127 | 8 Bit | Yes |
| USINT | 0 | 255 | 8 Bit | Yes |
| INT | -32,768 | 32,767 | 16 Bit | Yes |
| UINT | 0 | 65,535 | 16 Bit | Yes |
| DINT | -2,147,483,648 | 2,147,483,647 | 32 Bit | Yes |
| UDINT | 0 | 4,294,967,295 | 32 Bit | Yes |
| LINT | $-2^{63}$ | $2^{63}$-1 | 64 Bit | No |
| ULINT | 0 | $2^{64}$-1 | 64 Bit | No |
| REAL | 1.175494351e-38 | 3.402823466e+38 | 32 Bit | Yes |
| LREAL | 2.2250738585072014e-308 | 1.7976931348623158e+308 | 64 Bit | No |
| STRING | 1 character | 255 characters | 1 character = 1 byte | Yes |
| WSTRING | 1 character | 255 characters | 1 character = 1 word | Yes |
| TIME | - | - | 32 Bit | No |

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

Refer to Single Variable Definition for additional information on SoMachine/HMI data exchange.

## Using Array and Structure Elements for Data Exchange

You can use array and structure elements for data exchange between the controller side (SoMachine) and the HMI side (Vijeo-Designer). However, you cannot exchange whole arrays and structures at once.

For example:

- If `A` is an array, you can exchange an element of the array (`A[0],A[1],...,A[i]`) but not the entire array.
- The same rule applies to structure element, you can exchange an element of the structure (`StructureName.ElementName`) but not the entire structure.

# Variables Exchange

## Introduction

You can exchange variables with the XBTGC HMI Controller range between SoMachine and Vijeo-Designer by publishing them.

## Controller and HMI Data Exchange

For variable exchange between the controller and HMI parts, perform the following steps:

- Create variables in the controller part.
- Publish the variables by defining them as **Symbols** in the controller part. They are now available in the HMI part as SoMachine variables.

Refer to SoMachine Single Variable Definition *(see SoMachine, Programming Guide)* for additional information on how to publish variables.

Once symbols have been transferred toVijeo-Designer (the HMI part of your application), it is usually not necessary to make the transfer every time you call Vijeo-Designer. If you later add or modify symbols in your SoMachine application after having initially transfered the symbols, you must again transfer symbols to Vijeo-Designer.

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

After adding or modifying symbols shared between the XBTGC HMI Controller and other controllers, you must:

- Update the Vijeo-Designer application,
- Download the updated application into the XBTGC HMI Controller.

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

Refer to HMI Data Exchange *(see SoMachine, Programming Guide)* for additional information on how to exchange variables.

# Chapter 4
# Controller Memory Mapping

## Introduction

This chapter provides the maximum size of an application for a XBTGC HMI Controller, the size of the RAM , the located variables area and the libraries.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Memory Mapping | 28 |
| Controllers and HMI Address Mapping Differences | 29 |

# Memory Mapping

## Introduction

This section provides the RAM (Random Access Memory) size for each area of the XBTGC HMI Controller.

## XBTGC HMI Controller Memory

This table shows different types of areas and their corresponding size for the XBTGC HMI Controller memory allocated to CoDeSys control engine:

| Area | Element | Size (bytes) |
|---|---|---|
| System Area | System area reserved memory | 131072 |
| | System and diagnostic variables | |
| | Reserved input addresses (%I) | 256 |
| | Reserved output addresses (%Q) | 256 |
| | Retain variables(1)(2) | 16360 |
| | Persistent retain variables(2) | 2044 (2000 usable) |
| Application Area | Compiled control application | 1024000 |
| User Area(3) | Symbols | Dynamic allocation of 1228800 |
| | Variables | |
| | Libraries | |

**(1)** Not all of the 16360 bytes are available for the user application because some libraries may use retain variables.
**(2)** Retain variable data is held in SRAM requiring a battery backup.
**(3)** The symbols area size is not checked at build time. It is compiled with global data with the limit of 1228800 bytes.

# Controllers and HMI Address Mapping Differences

## Introduction

These paragraphs provide instructions for double words and bits addressing between controller and the XBTGC HMI Controller.

If you do not program your application to recognize the differences in address mapping between the controller and HMI parts, the controller and the HMI will not communicate correctly and it will be possible for incorrect values to be written to memory areas responsible for output operations.

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

Program your application to translate between the memory mapping used by the controller part and that used by the HMI part.

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

## Memory Data Exchange

When the controller and the XBTGC HMI Controller are connected, the data exchange uses simple word requests.

There is an overlap on simple words of the XBTGC HMI Controller memory while using double words but not for the controller memory:

<table>
<tr><th colspan="4">Controller Addressing</th><th></th><th colspan="4">HMI Addressing</th></tr>
<tr><td>%MX0.7...%MX0.0</td><td>%MB0</td><td rowspan="2">%MW0</td><td rowspan="2">%MD0</td><td rowspan="8">The double word is split into 2 simple words.</td><td rowspan="2"></td><td rowspan="4">%MD0</td><td rowspan="2">%MW0</td><td>%MW0:X7...%MW0:X0</td></tr>
<tr><td>%MX1.7...%MX1.0</td><td>%MB1</td><td>%MW0:X15...%MW0:X8</td></tr>
<tr><td>%MX2.7...%MX2.0</td><td>%MB2</td><td rowspan="2">%MW1</td><td rowspan="4">%MD1</td><td rowspan="4">%MW1</td><td>%MW1:X7...%MW1:X0</td></tr>
<tr><td>%MX3.7...%MX3.0</td><td>%MB3</td><td>%MW1:X15...%MW1:X8</td></tr>
<tr><td>%MX4.7...%MX4.0</td><td>%MB4</td><td>%MW2</td><td rowspan="4">%MD1</td><td rowspan="4">%MD2</td><td>%MW2</td><td>%MW2:X7...%MW2:X0</td></tr>
<tr><td>%MX5.7...%MX5.0</td><td>%MB5</td><td rowspan="2">%MW3</td><td>%MW2:X15...%MW2:X8</td></tr>
<tr><td>%MX6.7...%MX6.0</td><td>%MB6</td><td colspan="2">----------></td><td rowspan="2">%MW3</td><td>%MW3:X7...%MW3:X0</td></tr>
<tr><td>%MX7.7...%MX7.0</td><td>%MB7</td><td></td><td></td><td>%MW3:X15...%MW3:X8</td></tr>
</table>

In order to have a match between the XBTGC HMI Controller memory area and the controller memory area, the ratio between double words of XBTGC HMI Controller memory and the double words of controller memory is 2.

## Examples

The following gives examples of memory match for the double words:

- %MD2 memory area of the XBTGC HMI Controller corresponds to %MD1 memory area of the controller.
- %MD20 memory area of the XBTGC HMI Controller corresponds to %MD10 memory area of the controller.

The following gives examples of memory match for the bits:

- %MW0:X9 memory area of the XBTGC HMI Controller corresponds to %M1.1 memory area of the controller because the simple words are split in 2 distinct bytes in the controller memory.

# Chapter 5
# Tasks

## Introduction

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

The task types available are:

- Cyclic
- Freewheeling
- 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 system and task watchdog functions and explains their relationship to task execution.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Maximum Number of Tasks | 32 |
| Task Configuration Screen | 33 |
| Task Types | 36 |
| System and Task Watchdogs | 38 |
| Task Priorities | 39 |
| Default Task Configuration | 41 |

# Maximum Number of Tasks

## Maximum Number of Tasks

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

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

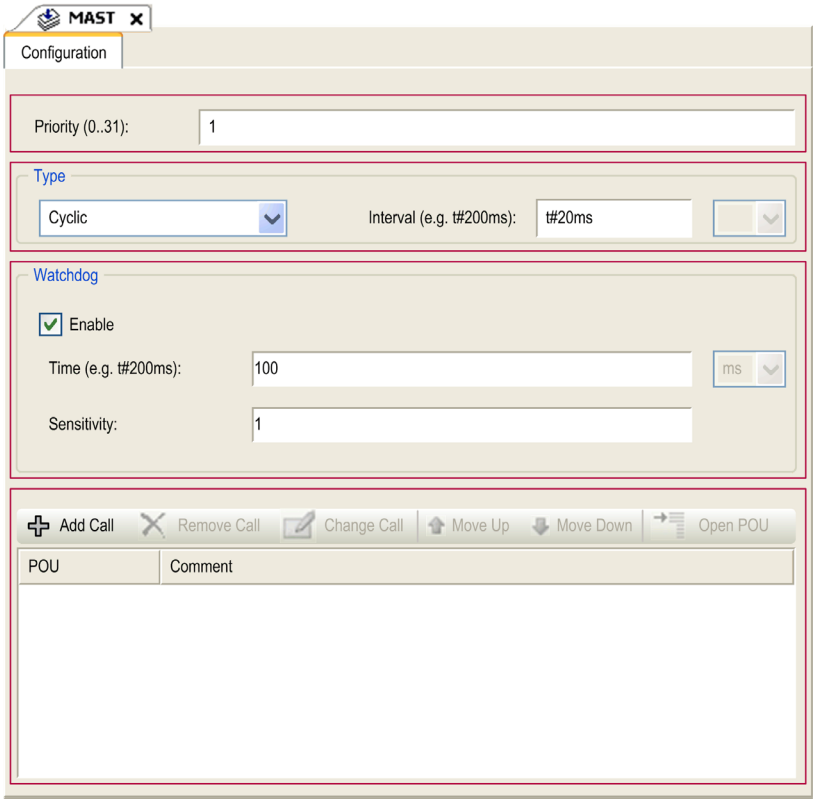
# Task Configuration Screen

## Screen Description

This screen allows you to configure the tasks. Double-click the task that you want to configure in the **Applications tree** tab to access this screen.

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

The **task configuration** window is composed of 4 parts:



This table describes the fields of the **Task Configuration** screen:

| Field Name | Definition |
|---|---|
| Priority | You can configure the priority of each task with a number between 0 and 31 (0 is the highest priority, 31 is the lowest).<br>Only one task at a time can be running. The priority determines when the task will run:<br>● a higher priority task will preempt a lower priority task<br>● tasks with same priority will run in turn (2 ms time-slice)<br>**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 information, Task Priorities *(see page 39)*. |
| Type | 4 types of task are available:<br>● Cyclic *(see page 36)*<br>● Freewheeling *(see page 37)*<br>● Event *(see page 37)* |
| Watchdog *(see page 38)* | To configure the watchdog, you must define 2 parameters:<br>● Time: enter the timeout before watchdog execution.<br>● Sensitivity: defines the number of expirations of the watchdog timer before the Controller stops in Exception mode. |
| POUs *(see SoMachine, Programming Guide)* | The list of **POU**s (Programming Organization Units) controlled by the task is defined in the task configuration window<br>● To add a POU linked to the task, use the command **Add Call** and select the POU in the **Input Assistant** editor.<br>● To remove a POU from the list, use the command **Remove Call**.<br>● To replace the currently selected POU of the list by another one, use the command **Change Call**.<br>● **POU**s are executed in the order shown in the list. To move the **POU**s in the list, select a **POU** and use the command **Move Up** or **Move Down**.<br>**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. |

## XBTGC HMI Controller Cycle Time Management

The XBTGC HMI Controller cycle time management is set with this configuration:

- 50% for the control
- 50% for the HMI application

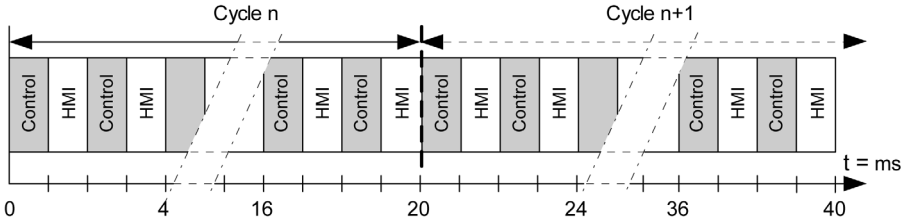
You should use a cycle time superior or equal to 20 ms.The period for the entire cycle must be a multiple of 4 ms (20, 24, 28, 32, 36 ms, and so on).

**NOTE:**

For XBTGC1100, 2120, 2230, and 2330 Embedded I/Os:

- There can be up to 4 ms latency between when an input gets a signal and when the controller gets this data.
- There can be up to 4 ms latency between when a variable is set and when the physical output actually changes state or value.

This diagram shows an example of cycle time management between the control and HMI parts. In this example, the cycle time is set to 20 ms:



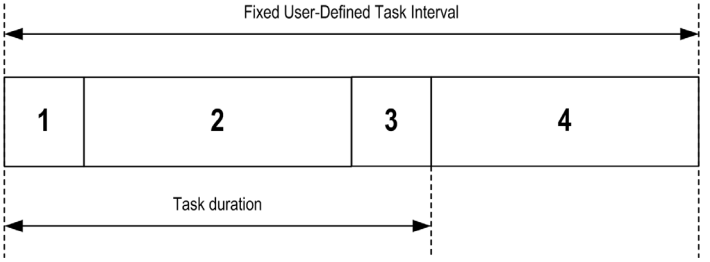
# Task Types

## 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 at a fixed cycle time using the Interval setting in the Type section of Configuration sub-tab for that task. Each Cyclic task type executes as follows:



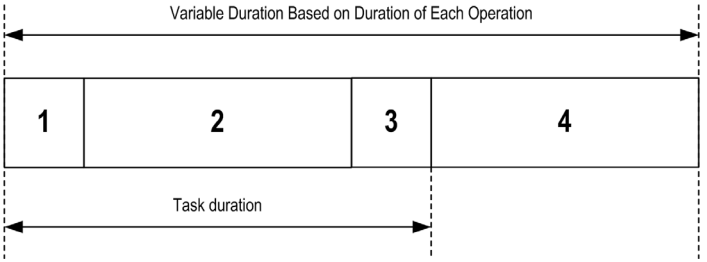

1. **Read Inputs:** The input states are written to the %I input memory variable and other system operations are executed.
2. **Task Processing:** The user code (POU, and so on) defined in the task is processed. The %Q output memory variable is updated according to your application program instructions but not written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variable is 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. For more information on I/O behavior, refer to Controller States Detailed Description *(see Magelis XBTGT, XBTGK HMI Controller, Programming Guide)*.
4. **Remaining Interval time:** The controller OS carries out system processing and any other lower priority tasks.

**NOTE:** If you define an insufficient period for a cyclic task, it will repeat immediately after the write of the outputs 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 task watchdog limits (if set by a user), generating a task watchdog exception.
For the XBTGC HMI Controller, system watchdog limits are not enforced.

**NOTE:** You can get and set the interval of a Cyclic Task by application using the **GetCurrentTaskCycle** and **SetCurrentTaskCycle** function.

## Freewheeling Task

A Freewheeling task does not have a fixed duration. Each Freewheeling task type executes as follows:




1. **Read Inputs:** The input states are written to the %I input memory variable 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 variable is updated according to your application program instructions but not written to the physical outputs during this operation.
3. **Write Outputs:** The %Q output memory variable is 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. For more information on I/O behavior, refer to Controller States Detailed Description *(see Magelis XBTGT, XBTGK HMI Controller, Programming Guide)*.
4. **System Processing:** The controller OS carries out system processing and any other lower priority tasks. The length of the system processing period is set to 30 % of the total duration of the 3 previous operations (4 = 30 % x (1 + 2 + 3)). In any case, the system processing period will not be lower than 3 ms.

## 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 preempted 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, select the **Event type** on the **Configuration** subtab and click the **Input Assistant** button  to the right of the **Event name** field. This will cause the **Input Assistant dialog** box to appear. In the **Input Assistant dialog** box, you navigate the tree to find and assign the `my_Var` variable.

# System and Task Watchdogs

## Introduction

2 types of watchdog functionality are implemented for the XBTGC HMI Controller:

- **Task Watchdogs**: Optional watchdogs that can be defined for each task. They are managed by your application program and are configurable in SoMachine.
- **Hardware Watchdog**: This watchdog is managed by the HMI controller main CPU. It is not configurable by the user.

## 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 the SoMachine Programming Guide.

# Task Priorities

## Introduction

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.

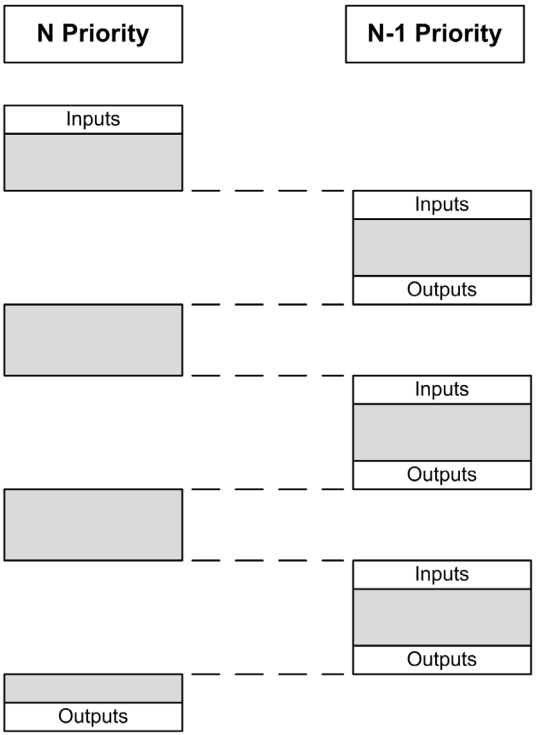
| ⚠ WARNING |
| --- |
| **UNINTENDED EQUIPMENT OPERATION** |
| Do not assign the same priority to different tasks. |
| **Failure to follow these instructions can result in death, serious injury, or equipment damage.** |

## Task Priority Recommendations

- Priority 0 to 24: Controller tasks. Assign these priorities to tasks with a high real-time requirement.
- Priority 25 to 31: Background tasks. Assign these priorities to tasks with a low real-time 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, an error is detected 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 of the XBTGC HMI Controller

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 39)* for more information on priority settings. Refer to System and Task Watchdogs *(see page 38)* 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. If you do so, SoMachine detects an error when you attempt to build the application, and you will not be able to download it to the controller.

# Chapter 6
# 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:

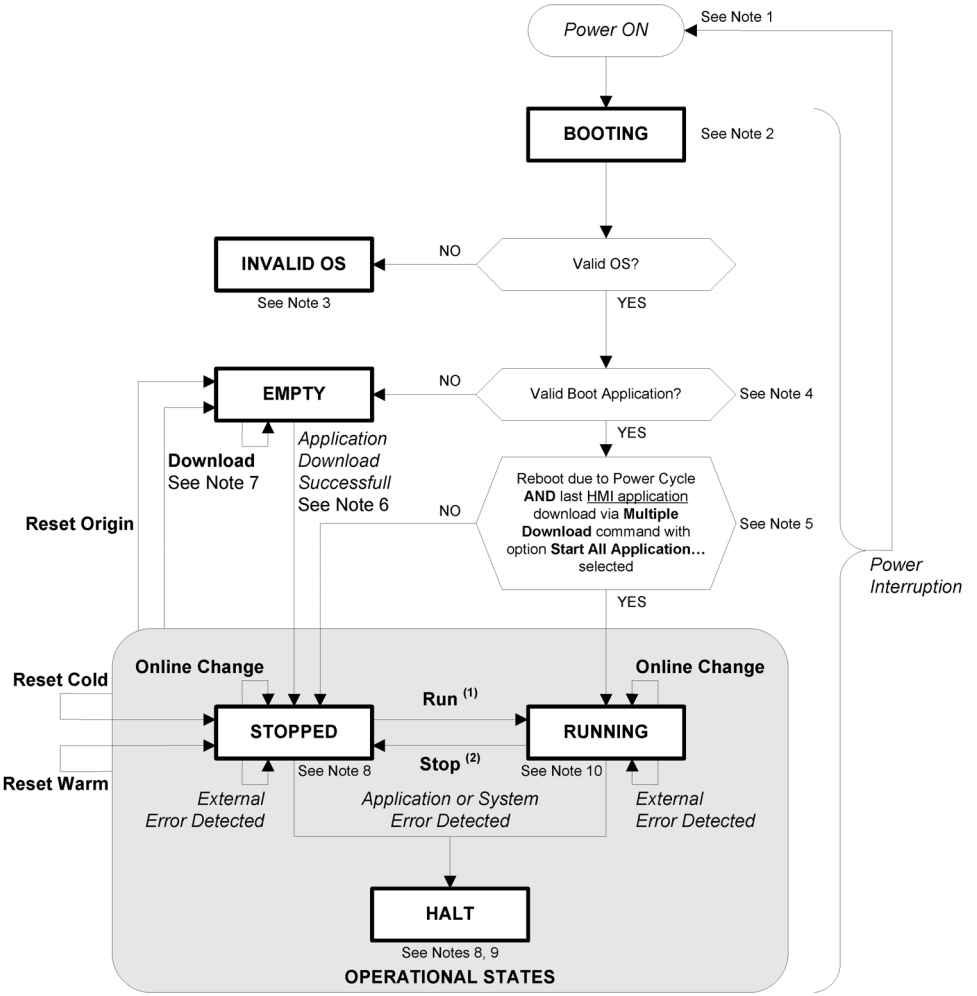

| Section | Topic | Page |
|---|---|---|
| 6.1 | Controller State Diagram | 44 |
| 6.2 | Controller States Description | 48 |
| 6.3 | State Transitions and System Events | 51 |

# Section 6.1
# 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 55)*.

(2) For details on RUNNING to STOPPED state transition, refer to Stop Command *(see page 55)*.

### Note 1

The Power Cycle (Power Interruption followed by a Power ON) deletes all output forcing settings. Refer to Controller State and Output Behavior *(see page 52)* 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 *(see page 52)* 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 48)* for further details on this exceptional condition.

# Section 6.2
# Controller States Description

## Controller States Description

### Introduction

This section provides a detailed description of the controller states.

**⚠ WARNING**

**UNINTENDED EQUIPMENT OPERATION**

- Never assume that your controller is in a certain controller state before commanding a change of state, configuring your controller options, uploading a program, or modifying the physical configuration of the controller and its connected equipment.
- Before performing any of these operations, consider the effect on all connected equipment.
- Before acting on a controller, always positively confirm the controller state by 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 XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT 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. |

| Controller State | Description |
|---|---|
| RUNNING with Breakpoint | This state is the same as the RUNNING state with the following exceptions:<br>● The task-processing portion of the program does not resume until the breakpoint is cleared.<br>For more information, refer to breakpoints management. |
| RUNNING with detection of an *External Error* | 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 49)* for an explanation of the behavior of outputs and field buses in this state. |
| STOPPED with detection of an *External Error* | 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.<br>This description is the same as for the STOPPED state with the following exceptions:<br>● 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:

- 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 6.3
## 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 | 52 |
| Commanding State Transitions | 55 |
| Error Detection, Types, and Management | 60 |
| Remanent Variables | 62 |

# Controller States and Output Behavior

## Introduction

The XBTGC 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:

- ControllerLockout Feature
- 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 states.

## Keep Current Values

You can select this option by choosing **Keep current values** in the **Behaviour for outputs in Stop** dropdown menu of the **PLC Settings** subtab of the **Controller Editor**. To access the Controller Editor, double-click **MyController** in the **Devices tree** and select **PLC settings** tab.

This output behavior applies in the STOPPED and HALT controller states. 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 48)* for more details on these variations.

### Set All Outputs to Default

You can select this option by choosing **Set all outputs to default** in the **Behaviour for outputs in Stop** dropdown menu of the **PLC Settings** subtab of the **Controller Editor**. To access the Controller Editor, double-click **MyController** in the **Devices tree** and select **PLC settings** tab.

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. Outputs are set to their user-defined default values, although the details of the output behavior vary greatly depending on the setting of the **Update IO while in stop** option and the actions commanded via configured fieldbusses. Refer to Controller States Description *(see page 48)* 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 transistor fast output: 0 Vdc
- For a transistor standard output: Z (High Impedance)
- For a relay output: Open

### Software Initialization Values

This output state applies when downloading or resetting the application.

It applies at the end of the download or at the end of a warm or cold reset.

The software initialization values are the initialization values of output 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/Watch menu.

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

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**, if supported by your controller, 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 apparently being 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 containted 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:

- SoMachine Online Menu: Select the **Start** command.
- By an HMI command using the PLC_W. q_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT 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 44)* 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 Run Command:

- SoMachine Online Menu: Select the **Stop** 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 XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT 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 Script: The 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 102)* and Reboot *(see page 97)* for further details.
- The controller is restarted into the STOPPED state automatically under certain conditions.

Refer to Controller State Diagram *(see page 44)* 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 XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT 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. All fieldbus communications are stopped and then restarted after the reset is complete.
8. All I/O are briefly reset to their initialization values and then to their user-configured default values.

For details on variables, refer to Remanent Variables *(see page 62)*.

## 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_wPLCControl and PLC_W. q_uiOpenPLCControl system variables of the XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT 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. All fieldbus communications are stopped and then restarted after the reset is complete.
8. All I/O are briefly reset to their initialization values and then to their user-configured default values.

For details on variables, refer to Remanent Variables *(see page 62)*.

## 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 62)*.

## 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 102)* 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 62)*.

**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, creates 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 44)*.
- USB memory key: Load Boot application file with the Download via File System method from Vijeo-Designer using a USB memory key connected to the controller USB port. The updated file is applied if the user accepts to install the new project when the Vijeo-Designer Runtimel prompts the user on the HMI screen. Refer to Saving your Application and Firmware on a USB Memory Key *(see page 102)* 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 62)*.

Effects of the USB memory key Download Command:

There are no effects 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 97)*.

# Error Detection, Types, and Management

## Detected Error Management

The controller manages 3 types of detected errors:

- external detected errors
- application detected errors
- system detected errors

The following table describes the types of errors that may be detected:

| Type of Error Detected | Description | Resulting Controller State |
|---|---|---|
| External Error Detected | 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:<br>● A connected device reports an error to the controller<br>● The controller detects an error with an external device whether or not it reports an error, for example when the external device is communicating but not properly configured for use with the controller<br>● The controller detects an error with the state of an output<br>● The controller detects a loss of communication with a device<br>● The controller is configured for a module that is not present or not detected<br>● The boot application in Fash memory is not the same as the one in RAM.<br>Examples:<br>● output short circuit<br>● missing expansion module<br>● communication lost<br>● etc. | RUNNING with External Error Detected<br>Or<br>STOPPED with External Error Detected |
| Application Error Detected | An application error is detected when improper programming is encountered or when a task watchdog threshold is exceeded.<br>Examples:<br>● task (software) watchdog exception<br>● execution of an unknown function<br>● etc. | HALT |

| Type of Error Detected | Description | Resulting Controller State |
|---|---|---|
| System Error Detected | 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.<br>Examples:<br>● exceeding the defined size of an array<br>● etc. | BOOTING → EMPTY |

**NOTE:** Refer to the XBT PLCSystem library *(see Magelis XBTGC, XBTGT, XBTGK HMI Controller, System Functions and Variables, XBT PLCSystem Library Guide)* for more detailed information on diagnostics.

**NOTE:** For XBTGC HMI Controller, System (hardware) watchdog overflow detection is not supported.

# Remanent Variables

## Remanent Variables

Remanent variables can retain their values in the event of power outages, reboots, resets, and application program downloads. There are multiple types of remanent variables, declared individually as "retain" or "persistent", or in combination as "retain-persistent".

**NOTE:** For this controller, variables declared as persistent have the same behavior as variables declared as retain-persistent.

The following table describes the behavior of remanent variables in each case:

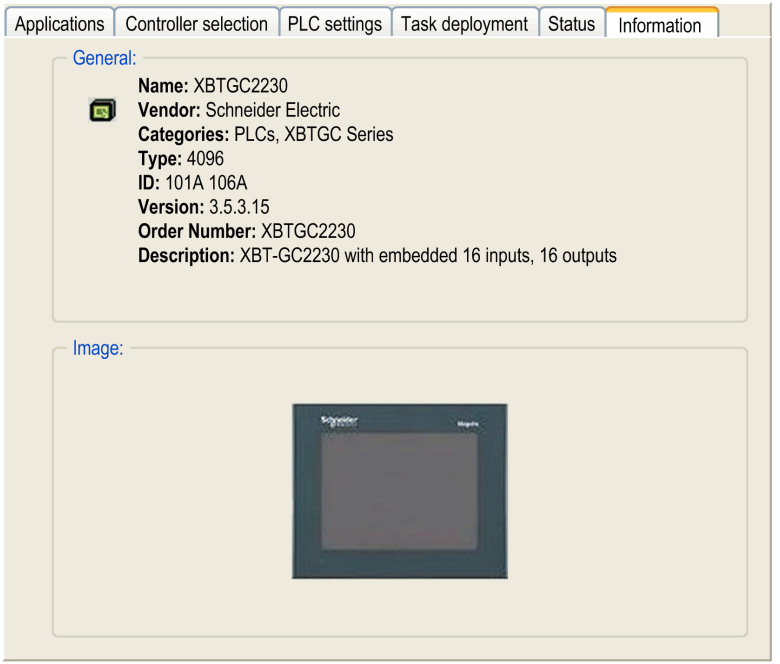
| Action | VAR | VAR RETAIN | VAR PERSISTENT and RETAIN-PERSISTENT |
|---|---|---|---|
| Online change to application program | X | X | X |
| Stop | X | X | X |
| Power cycle | - | X | X |
| Reset warm | - | X | X |
| Reset cold | - | - | X |
| Reset origin | - | - | - |
| Download of application program | - | - | X |
| **X** The value is maintained<br>**-** The value is reinitialized | | | |

# Chapter 7
# Controller Configuration

## Device Editor

### Introduction

Configure and monitor your XBTGC HMI Controller using the **Device Editor**. The following screen-shot shows the **Information** tab of the Device Editor window:




### XBTGC HMI Controller Device Editor Window

To open the XBTGC HMI Controller Device Editor, double-click **MyController** node.

## Tabs Description

This table provides a description of the tabs available from the Device Editor window:

| Tab | Description |
|---|---|
| **Applications** | Shows the applications currently running on the controller and allows removing applications from the controller (not available for expansion modules). |
| **Controller selection** | Allows configuring the parameters for the communication between the controller and the programming system. |
| **PLC Settings** | Allows configuring the fallback of the outputs. |
| **Task deployment** | Shows a table with inputs/outputs and their assignment to the defined tasks. |
| **Status** | Displays device-specific status and diagnostic messages. |
| **Information** | Displays general information about the device (name, description, provider, version, image). |

# Chapter 8
# Embedded I/O Configuration

## Embedded I/O Configuration Editor

### Introduction

Configure and monitor the I/Os of your controller using the Embedded I/O Configuration Editor. This table shows the number of standard I/Os for each XBTGC HMI Controller:

| XBTGC HMI Controller | Number of Digital Inputs | Number of Digital Outputs |
|---|---|---|
| XBTGC1100 | 12 | 6 |
| XBTGC2120 | 16 | 16 |
| XBTGC2230 | 16 | 16 |
| XBTGC2330 | 16 | 16 |

The standard embedded inputs are:
- For the XBTGC1100: I0 to I11
- For the XBTGC2120: I0 to I15
- For the XBTGC2230/XBTGC2330: I0 to I15

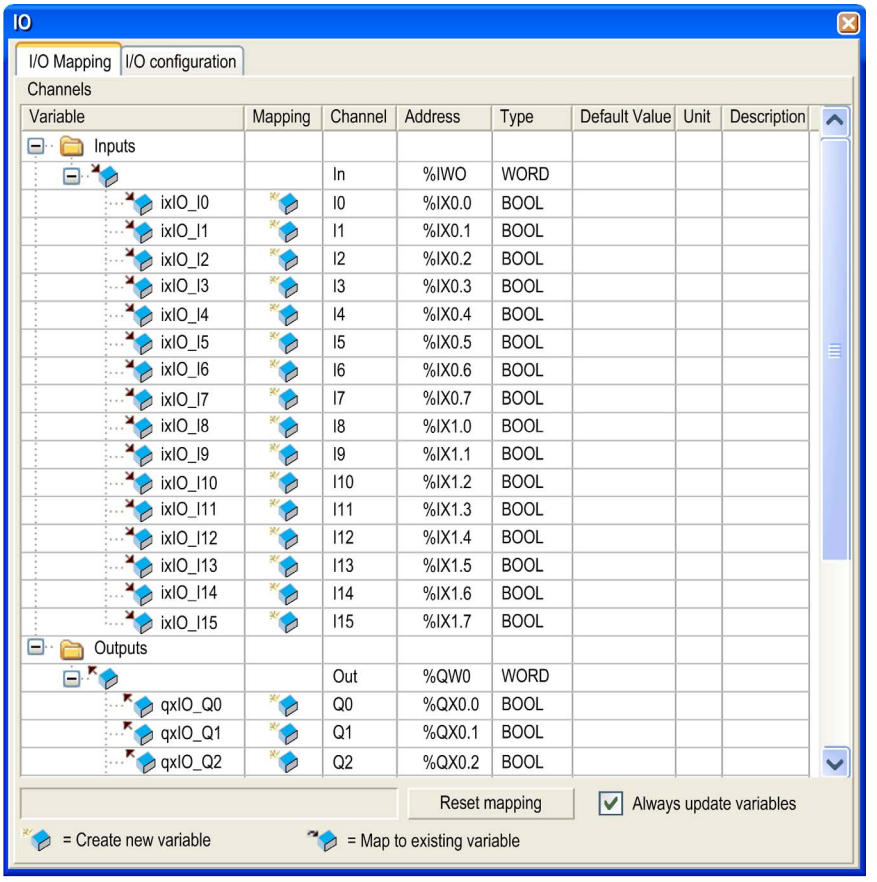
The standard embedded outputs are:
- For the XBTGC1100: Q0 to Q5
- For the XBTGC2120: Q0 to Q15
- For the XBTGC2230/XBTGC2330: Q0 to Q15

### Accessing the Embedded I/O Configuration Editor

To access the I/O configuration window, double-click **MyController → Embedded Functions → IO**.

## I/O Mapping Tab

Configure the I/O mapping via the I/O mapping tab:




**NOTE:** For more information on **I/O Mapping** tab, refer to I/O Mapping.

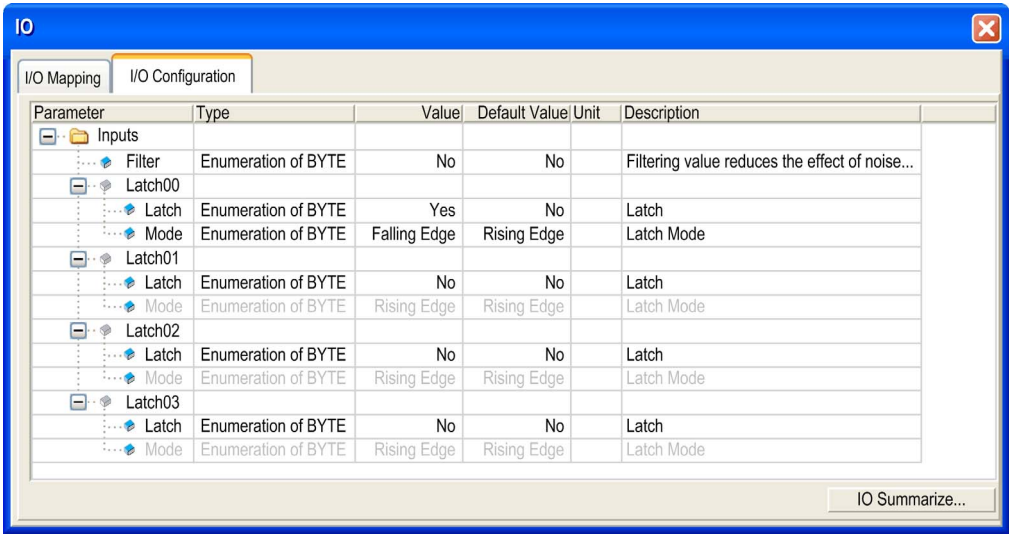
## I/O Mapping Tab Parameters

You can assign different parameter to map the I/Os:

| Parameters | Description |
|---|---|
| **Mapping** | Method for creating or mapping an existing variable |
| **Channel** | Channel used by the variable |
| **Address** | Address of the variable |
| **Type** | Type of the variable |
| **Default Value** | Value of the variable by default |
| **Unit** | Unit of the variable |
| **Description** | Brief description on the I/O; for example: Fast Input |

## Configuration Tab

Configure your embedded inputs via the configuration tab:



## Configuration Tab Parameters

You can define a global input filter:

| Parameter | Value | Default Value | Description | Constraint |
|---|---|---|---|---|
| **Filter** | No<br>1.5 ms<br>4 ms<br>12 ms | No | The filtering value reduces the effect of electromagnetic noise on a controller input. | Enabled if Latch and Event are disabled.<br>In all other cases, this parameter is disabled and its value is No. |
| **Latch** | No/Yes | No | Latching allows incoming pulses with amplitude widths shorter than the controller scan time to be captured and recorded. | You can configure 4 latches. |
| **Mode** | Rising Edge<br>Falling Edge | Rising Edge | Configures the triggering mode: rising or falling edge. | – |

# Chapter 9
# Special I/O Configuration

## Introduction

This chapter describes how local I/Os can be configured as special I/Os.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Local and Special I/O Overview | 70 |
| Special I/O Configuration Possibilities | 72 |
| I/O Summary | 76 |

# Local and Special I/O Overview

## Introduction

The XBTGC HMI Controller supports the following local I/Os:

| Controller | Inputs | Outputs |
|---|---|---|
| XBTGC1100 HMI Controller | 12 hardware inputs | 6 hardware outputs |
| XBTGC2120 HMI Controller<br>XBTGC2230 HMI Controller<br>XBTGC2330 HMI Controller | 16 hardware inputs | 12 hardware outputs |

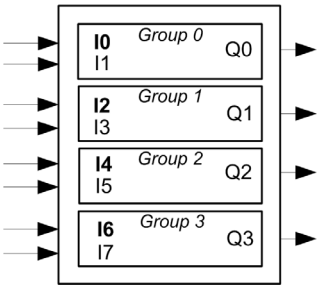
## Special I/O Types

The local I/O can be configured as special I/O. Special I/Os include:

- High Speed Counter (HSC) *(see Magelis XBTGC HMI Controller, High Speed Counting, XBTGC HSC Library Guide)*
- Pulse Train Output (PTO) *(see Magelis XBTGC HMI Controller , Pulse Train Output, Pulse Width Modulation, XBTGC PTOPWM Library Guide)*
- Pulse Width Modulation (PWM) *(see Magelis XBTGC HMI Controller , Pulse Train Output, Pulse Width Modulation, XBTGC PTOPWM Library Guide)* output
- Pulse Latch Input (PLI) *(see Magelis XBTGC HMI Controller , Pulse Train Output, Pulse Width Modulation, XBTGC PTOPWM Library Guide)*

## Special I/O Configuration

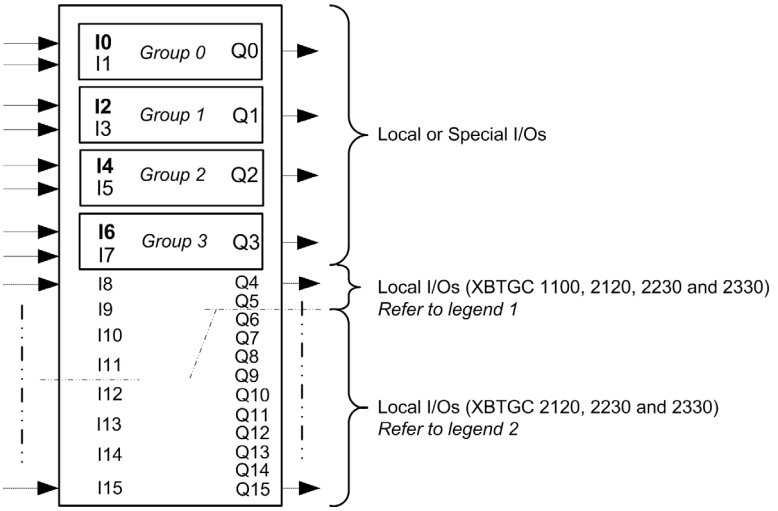
Special I/Os are configured in four Groups. Each Group has two inputs ($I_n$ and $I_{n+1}$ of Group *n*) and one output ($Q_n$ of Group *n*), as shown in the diagram below:




**NOTE:** Any remaining I/Os can be configured as normal I/O. *(see page 71)*.

## Local I/Os and Special I/Os Configuration

The following diagram provides the local and special I/Os configuration:



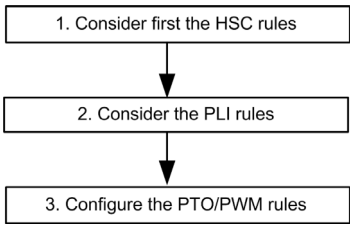

Legend

1 The local I/Os for the XBTGC1100 HMI Controller are from I8 to I11 and from Q4 to Q5.

2 The local I/O for the XBTGC2120 HMI Controller, the XBTGC2230 HMI Controller and the XBTGC2330 HMI Controller are from I8 to I15 and from Q4 to Q15.

## Special I/Os Configuration Order

When configuring special I/Os, follow the order shown in the diagram below:




The special I/Os configuration depends on the number and type of HSC required. There are 3 cases:

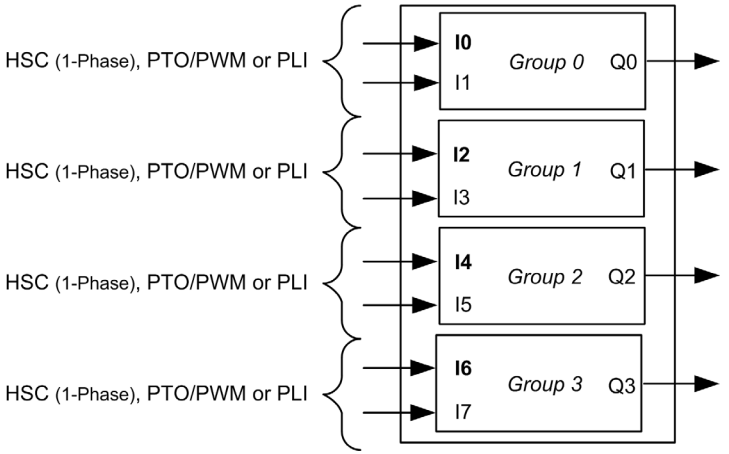
- **Case 1**: *(see page 72)* No HSC is required, or only 1-Phase HSC is required (also referred to No 1-Phase HSC)
- **Case 2**: *(see page 73)* One 2-Phase HSC is required
- **Case 3**: *(see page 74)* Two 2-Phase HSC are required

See more specific information in the HSC Configuration chapter *(see Magelis XBTGC HMI Controller, High Speed Counting, XBTGC HSC Library Guide)*.

# Special I/O Configuration Possibilities

## Case 1: 1-Phase HSC Combination

All Groups can be configured independently as HSC, PLI or PTO/PWM:



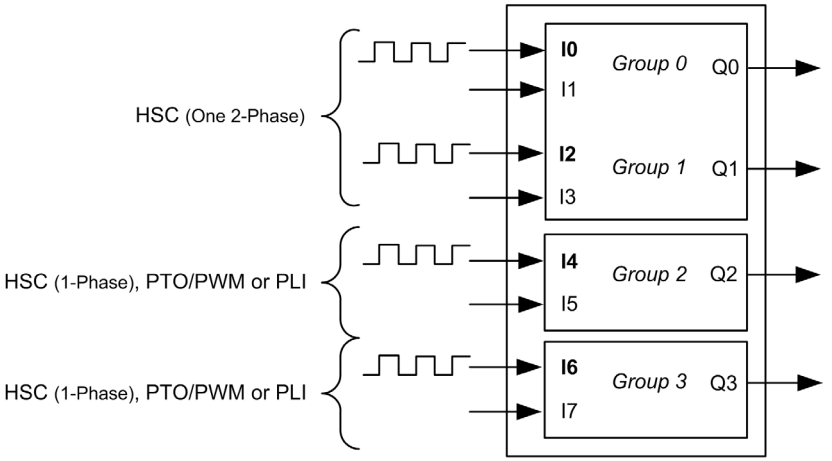

These Groups can provide the combinations shown in the following table:

| Main Functions | $I_{(2n)}$ | $I_{(2n+1)}$ | $Q_{(n)}$ |
|---|---|---|---|
| 1-Phase HSC Input | 1-Phase HSC Input | Normal Input or<br>Preload or<br>Prestrobe | Normal Output or<br>Synchronized Output |
| Normal I/O, PWM or PTO | Normal Input | Normal Input | Normal Output or<br>PWM or<br>PTO |
| PLI | Pulse Latch Input | Normal Input | Normal Output |

**NOTE:** *n* is the Group number starting from 0 to 3 (HSC0*n*/PTO0*n*/Latch0*n*) where $I(_{2n})$, $I(_{2n+1})$ and $Q(_{n})$ are the inputs and output respectively of the Group *n*.

### Case 2: One 2-Phase HSC Combination

Group 0 and Group 1 form a 2-Phase HSC, the others can be configured as HSC, PLI or PTO/PWM:




For this combination, Group 0 (HSC00) and Group 1 (HSC01) are combined to form a 2-Phase HSC. The following tables show the combinations available:

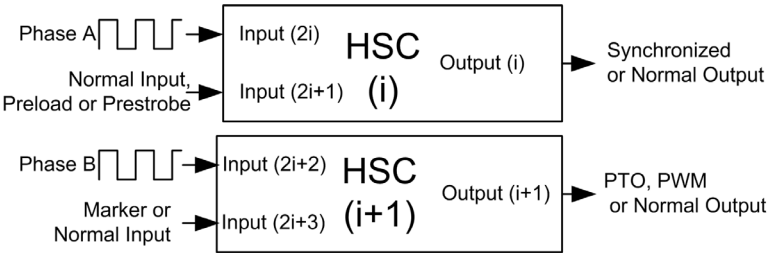
| I0 | I1 | Q0 |
|---|---|---|
| Counter 1A | Normal Input or<br>Preload or<br>Prestrobe | Normal Output or<br>Synchronized Output |

| I2 | I3 | Q1 |
|---|---|---|
| Counter 1B | Marker Input or<br>Normal Input | Normal Output or<br>PWM or<br>PTO |

**NOTE:** Group 2 and Group 3 (HSC0*n*/PTO0*n*/Latch0*n*) follow the rules of the 1-Phase HSC combination.

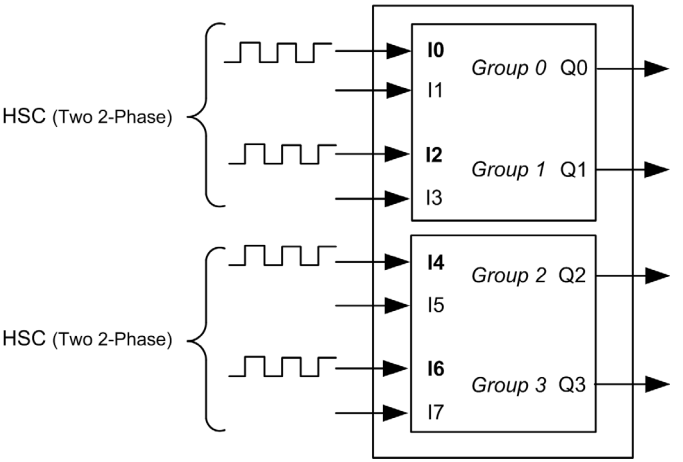
One 2-Phase HSC combinations summary:

- The PLI function is not available on any input of the Group.
- The PWM and PTO functions are available on the second output of the second HSC in the Group.
- The Synchronized Outputs are available on the output of the first HSC in the Group.




## Case 3: Two 2-Phase HSC Combination

The following diagram shows the Two 2-Phase HSC Combination:



For this combination, Group 0 (HSC00) and Group 1 (HSC01) are combined to form a 2-Phase HSC. Group 2 (HSC02) and Group 3 (HSC03) form another 2-Phase HSC. The following tables show the functions available:

| I0 or I4 | I1 or I5 | Q0 or Q2 |
|---|---|---|
| Counter 1A | Normal Input or<br>Preload or<br>Prestrobe | Normal Output or<br>Synchronized Output |

| I2 or I6 | I3 or I7 | Q1 or Q3 |
|---|---|---|
| Counter 1B | Normal Input or<br>Marker Input | Normal Output or PWM or<br>PTO |

Two 2-Phase HSC combinations summary:

- The PLI function cannot be used with two 2-Phase HSC configuration.
- The PWM and PTO functions are available on the second output of the second HSC in the Group 1 (HSC01) or Group 3 (HSC03).
- The Synchronized Output is available on the output of the first HSC in the Group 0 (HSC00) and on the output of the third HSC in the Group 2 (HSC02).

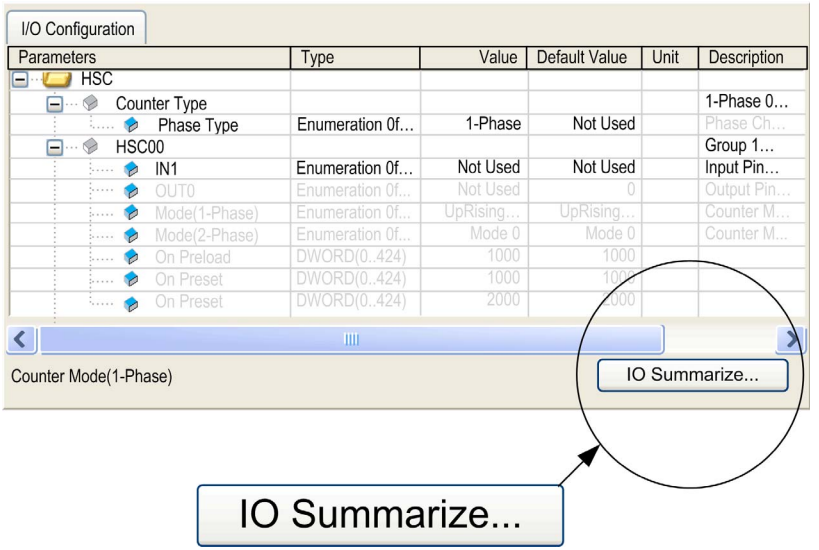
# I/O Summary

## Overview

The I/O summary shows the current I/O pin configuration for I/O nodes as HSC, PTO/PWM and PLI.

To access the I/O summary, click the **IO Summarize...** button available on the configuration screen of each function.

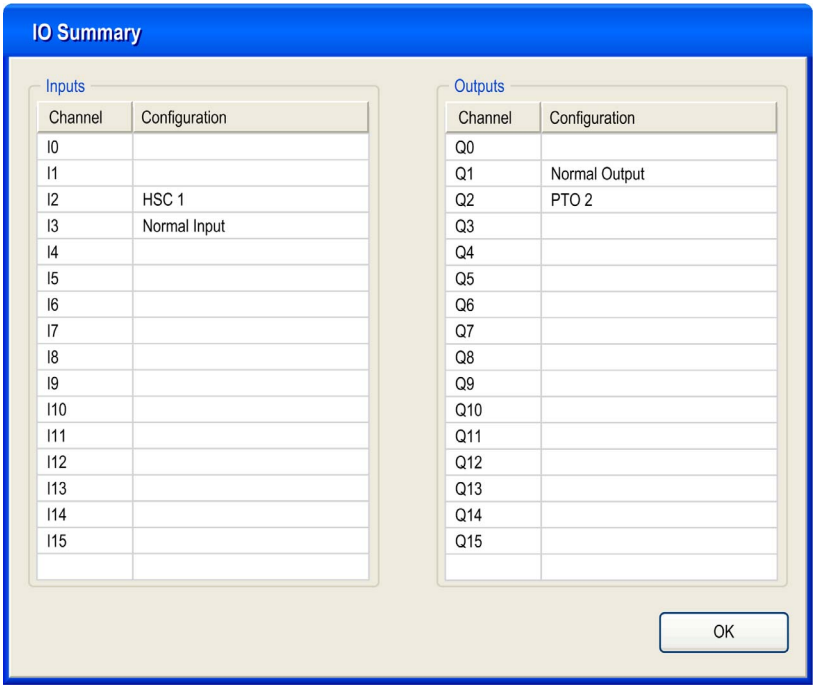
This picture shows, as an example, the HSC **IO Summary**:




**NOTE:** The **IO Summarize...** button is common to all functions and can be accessed from the configuration screen of each function: HSC, PTO/PWM and PLI.

## I/O Summary Window

Click the **IO Summarize** button to display this window:



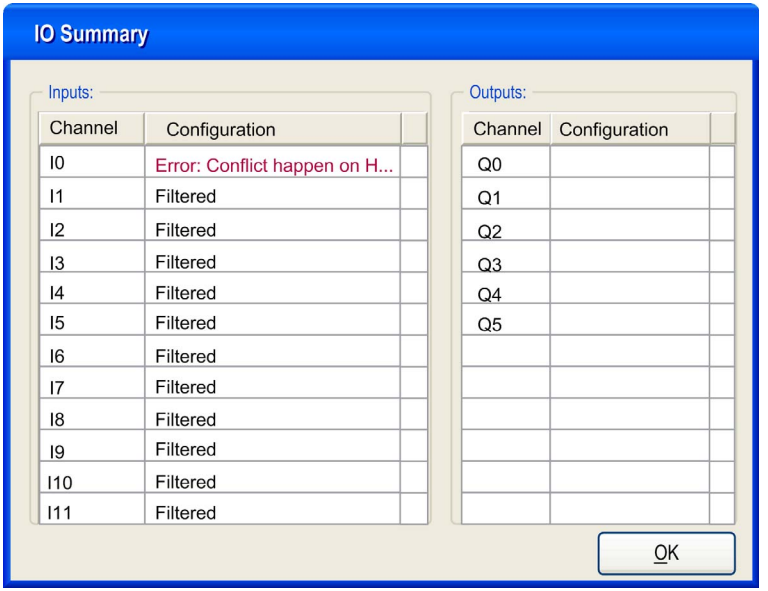

## I/O Summary Messages

If an I/O setting inconsistency is detected, the **Configuration** column from the **IO Summary** provides 2 types of messages:

- **Error: Conflict happened on HSC and IO settings**
- **Error: Conflict happened on HSC and PTO/PWM settings**

## Example of I/O Summary

This example shows the **IO Summary** window when I/O is configured as a standard input, with a Prestrobe input including a detected error message:



# Chapter 10
# Expansion Modules Configuration

## Introduction

This chapter provides information on configuring I/O expansion modules inputs and outputs.

## What Is in This Chapter?

This chapter contains the following sections:


| Section | Topic | Page |
|---|---|---|
| 10.1 | I/O Configuration | 80 |
| 10.2 | Digital I/O Modules | 81 |
| 10.3 | Analog I/O Modules | 82 |

# Section 10.1
# I/O Configuration

## General Considerations

### XBTGC HMI Controller Maximum Hardware Configuration

For more information about I/O expansion modules, please refer to:

- Adding Expansion Modules *(see page 19)* when creating a project
- *I/O Expansion Modules (see Magelis XBTGC HMI Controller, Hardware Guide)* for the list of expansion modules and their allowed combinations
- *Digital Input and Output Expansion Modules (see SoMachine, Introduction)* for the lists of supported digital modules
- *TM2 Digital I/O Expansion Modules (see Modicon TM2, Digital I/O Modules, Hardware Guide)* for the digital modules hardware implementation
- *Analog Input and Output Expansion Modules (see SoMachine, Introduction)* for the lists of supported analog modules
- *TM2 Analog I/O Expansion Modules (see Modicon TM2, Analog I/O Modules, Hardware Guide)* for the Analog modules hardware implementation

# Section 10.2
# Digital I/O Modules

## TM2 Digital I/O Modules

### Refer to

Refer to *I/O Expansion Modules Configuration (see Modicon TM2, Digital I/O Modules, Hardware Guide)*, for additional information on how to configure the TM2 digital I/O modules.

# Section 10.3
# Analog I/O Modules

## TM2 Analog I/O Modules

### Refer to

Refer to *I/O Expansion Modules Configuration (see Modicon TM2, Analog I/O Modules, Hardware Guide)*, for additional information on how to configure the TM2 analog I/O modules.

# Chapter 11
# Ethernet Configuration

## IP Address Configuration

### Introduction

Setting up an Ethernet connection and IP address configuration with the HMI controllers is accomplished with Vijeo-Designer.

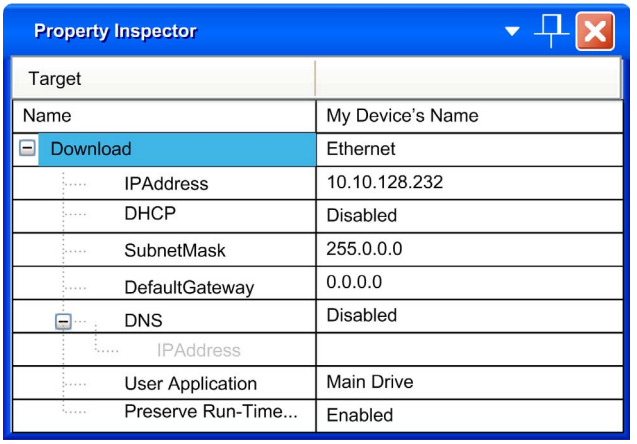
There are two ways to assign the IP address of the controller with Vijeo-Designer:

- DHCP server
- Fixed IP address

**NOTE:** If the above addressing modes are not operational, the PLC starts with a *default IP address* *(see page 84)* derived from its MAC address.

### Ethernet Configuration

For the HMI controller, the Ethernet configuration is done via the Vijeo-Designer **Property Inspector** window:




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

The following table briefly explains the different parameters needed for setting up an Ethernet configuration:

| Element | Description |
|---|---|
| **Download** | Choose the project download method you want in the drop-down menu list. When configuring Ethernet connection, select **Ethernet**. The project download methods are:<br>● Ethernet<br>● File System<br>● USB<br>● SoMachine |
| **IP Address** | IP address of the controller. |
| **DHCP** | When DHCP is:<br>● **Enable**: The controller automatically retrieves an IP address from a DHCP server.<br>● **Disabled**: The controller uses a static IP address. |
| **SubnetMask** | When using a static IP setting, provide the subnet mask of your network. |
| **DefaultGateway** | When using a static IP setting, provide the default gateway of your network. |
| **DNS** | **Enable** DNS to use domain names instead of IP addresses. |
| **DNS IP Address** | When using DNS, provide the IP address for the DNS server. |

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

## 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 device's MAC address.

The default subnet mask is 255.0.0.0.

**NOTE:** A MAC address has an 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.

# Chapter 12
# CANopen Configuration

## Introduction

This chapter describes how to configure the CANopen network interface of the XBTGC HMI Controller.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| CANopen Interface Configuration | 86 |
| CANopen Optimized Manager | 88 |
| CANopen Remote Devices | 89 |

# CANopen Interface Configuration

## XBTGC HMI Controller Maximum Hardware Configuration

The hardware configuration requirements for the XBTGC HMI Controller are:

- Only one CANopen expansion module **or** a set of I/O expansion modules can be attached to the XBTGC HMI Controller. It is **not** physically possible to have a CANopen module and an I/O expansion module together.
- Up to 16 CANopen remote devices can be connected to the CANopen Master Unit.

## XBTGC HMI Controller Software Requirements

The maximum number of Received PDO RPDO is 32.

The maximum number of Transmitted PDO TPDO is 32.

## Adding the CANopen Expansion Modules

When adding a XBTZGCCAN CANopen expansion module to the XBTGC HMI Controller, the CANbus node is automatically created. Additional CANopen devices can be added to the manager.

**⚠ 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.**

To add a CANopen expansion module to your project, select the **XBTZGCCAN** expansion module in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes .

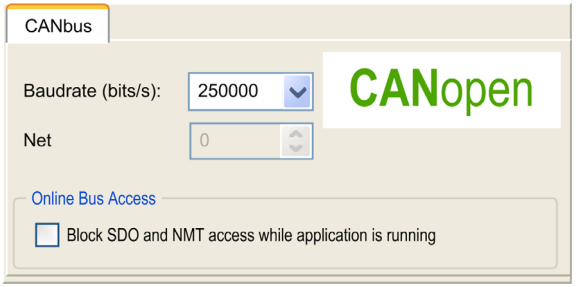
For more information on adding a device to your project, refer to:

• Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

## Baudrate Configuration

This table provides the procedure for accessing the CANopen baudrate configuration screen:

| Step | Action |
|---|---|
| 1 | Double-click **CANbus → CAN** in the **Devices tree**.<br>**Result:** The **CANbus** configuration screen appears:<br> |
| 2 | Select the **CANbus** tab. |
| 3 | Configure the baudrate using the **Baudrate (bits/s)** menu list. By default, the value is set to 250,000 bit/s. |
| 4 | Configure the net using the **Net** menu list. By default, the value is set to 0. |
| 5 | Configure the online bus access by clicking **Block SDO and NMT access while application is running**. By default, the online bus access is activated. |

## CANopen Network Manager

Configure the CANopen **Network_Manager** when using CANopen:

| Element | Description |
|---|---|
| **CANopen_Optimized**-Network_Manager | Used to support the CANbus configuration by internal functions (1). |
| (1) Refer to *CANopen Optimized Manager (see page 88)* for additional information on the configuration. | |

# CANopen Optimized Manager

## CANopen Optimized Manager Configuration Screen

You can access the **CANopen_Optimized** manager configuration screen by double-clicking the **CANopen_Optimized** node from the **Device tree**.

For more information on CANopen managers, refer to Adding Communication Managers.

# CANopen Remote Devices

## Remote Devices Available with CANopen

This list shows the remote devices available with CANopen and supported by SoMachine:

- Variable speed drives such as Altivar.
- Servo drives such as Lexium.
- Integrated drives such as ILA1F, ILE1F or the ILS1F.
- Opto-electronic encoders such as the Osicoder.
- Configurable safety controllers such as the Preventa.
- Stepper motor drives.
- Motor management and protection systems such as TeSysT.
- Starter controllers such as TeSysU.
- Distributed I/Os such as TVD_OTB.

**NOTE:** Other CANopen devices can be added using their electronic data sheet (EDS) files.

Refer to *Supported Devices (see SoMachine, Introduction)* for additional information.

For additional information on these remote devices, refer to external devices documentation available on Schneider Electric website.

## Adding a Remote Device to the Controller

To add a remote device to your controller, select it in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

• Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

## CANopen Remote Device Configuration Screen

Access the remote device configuration screen by double-clicking the device from the **Devices tree**. Refer to CANopen remote device part from the CoDeSys Online-Help for more information.

# Chapter 13
# Serial Line Configuration

## Introduction

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

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Serial Line Configuration | 92 |
| SoMachine Network Manager | 94 |
| Modbus Manager | 95 |

# Serial Line Configuration

## Introduction

The serial line configuration window allows configuration of the serial line parameters (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.

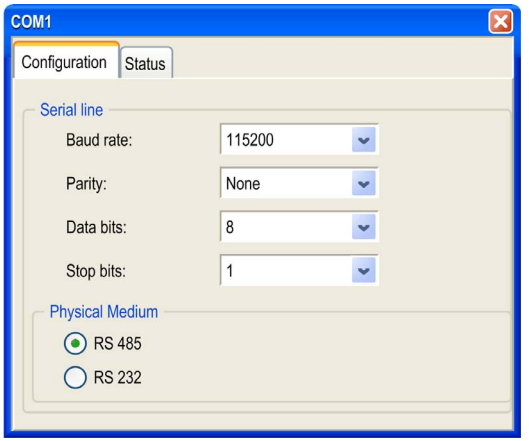
In an active Modbus configured serial line, if a new controller is connected or if a controller firmware is updated, this can cause the other devices available on the serial line to stop communicating.

Verify 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.** |

## Serial Line Configuration Window

Double-click **COM1** in the **Devices tree** to access the serial line configuration window. These parameters must be identical for each Modbus device on the link:



This table provides the description of each parameter:

| Parameter | Initial Values | Range | Description |
|---|---|---|---|
| **Baud rate** | 115.2 Kbauds | 1.2...115.2 Kbauds | Transmission speed |
| **Parity** | None | • None<br>• Odd<br>• Even | Used for invalid events detection |
| **Data bits** | 8 | • 7<br>• 8 | Number of bits for transmitting data |
| **Stop bits** | 1 | • 1<br>• 2 | Number of stop bits |
| **Physical Medium** | RS 485 | • RS485<br>• RS232 | Specify the medium to use |

## Network Manager

The SoMachine-Network_Manager is automatically added to your project configuration. You can configure 2 types of **Network_Manager** with the serial line:

| Element | Description |
|---|---|
| SoMachine-Network_Manager | Used when a XBTGC HMI Controller device is used, or when the Serial Line is also used for programming (1) the controller. |
| Modbus_Manager | Used for Modbus RTU or ASCII protocol in master or slave mode (2). |
| (1) Refer to SoMachine *Network_Manager (see page 94)* for additional information on the configuration.<br>(2) Refer to *Modbus Manager (see page 95)* for additional information on the configuration. | |

**NOTE:** When using the SoMachine-Network_Manager you can dowload your application to any devices connected to it.

# SoMachine Network Manager

## Adding a SoMachine Network Manager

To add a SoMachine Network Manager to your project, select the **SoMachine-Network Manager** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

- Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*
- Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

**NOTE:** The Serial Line link does not support both Modbus and SoMachine protocols at the same time.

# Modbus Manager

## Adding a Modbus Manager

To add a Modbus Manager to your project, select **Modbus_Manager** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

For more information on adding a device to your project, refer to:

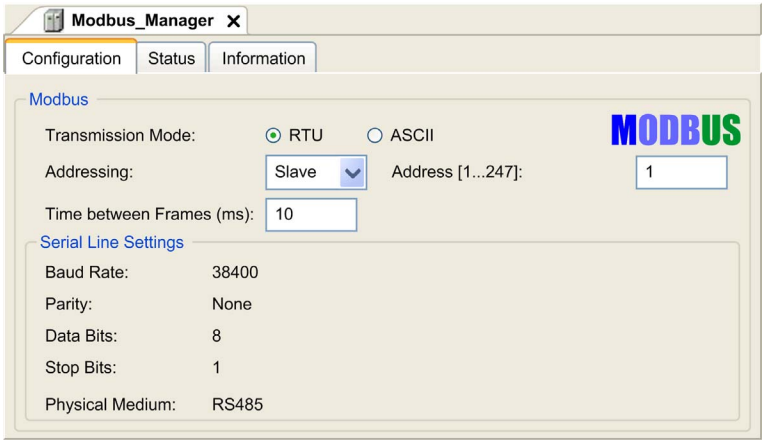
• Using the Drag-and-drop Method *(see SoMachine, Programming Guide)*

• Using the Contextual Menu or Plus Button *(see SoMachine, Programming Guide)*

**NOTE:** The Serial Line link does not support both Modbus and SoMachine protocols at the same time.

## Modbus Manager Configuration Window

Double-click **Modbus_Manager** in the **Devices tree** to access the Modbus manager **Configuration** tab:




This table provides the description of Modbus parameters:

| Element | Description |
|---|---|
| **Modbus** | |
| **Addressing** | Specify the device type:<br>● Master |
| **Address [1...247]** | Modbus address of the device if device type is set to Slave. For HMI controllers, this field is not used. |
| **Time between Frames (ms)** | Time required to avoid bus-collision<br>Set this parameter identical for each Modbus device on the link. |

| Element | Description |
|---|---|
| **Serial Line Settings** | |
| **Baud Rate** | Transmission speed |
| **Parity** | Used for error detection |
| **Data Bits** | Number of bits for transmitting data |
| **Stop Bits** | Number of stop bits |
| **Physical Medium** | Medium currently used, it can be either:<br>● RS-485, or<br>● RS-232 |

# Chapter 14
# Managing Online Applications

## Connecting the Controller to a PC

### Application Transfer

To transfer and run applications, connect your XBTGC HMI Controller to a PC with a properly installed version of SoMachine. To transfer an application, use Ethernet, serial link, USB cables, or a USB memory key.

| *NOTICE* |
| --- |
| **POSSIBLE ELECTRICAL DAMAGE TO CONTROLLER COMPONENTS**<br>Connect the communication cable to the PC before connecting it to the controller.<br>**Failure to follow these instructions can result in equipment damage.** |

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

### Automatic Reboot After Application Transfer

The XBTGC HMI Controller 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 cables or a USB memory key), the firmware update is done automatically. As a matter of good practice, always have a backup of your application/firmware combination on a USB memory key *(see page 101)*. Archive your application properly with the versions of SoMachine it was created and maintained under.

## USB Cables Requirements

To connect the controller to your PC, specific USB cables are required as shown in this table:

| Product Name | Reference | Description |
|---|---|---|
| USB Transfer Cable | XBTZG935 | Download project data created with the window Editor via the USB interface from the XBTGC unit. |
| USB Front Cable | XBTZGUSB | Extension cable attaching USB port to front panel. |
| USB Front Cable | XBTZGUSBB | Extension cable attaching USB port to front panel. |
| USB Programming Cable | TCSXCNAMUM3P | Extension cable attaching USB port to front panel. |

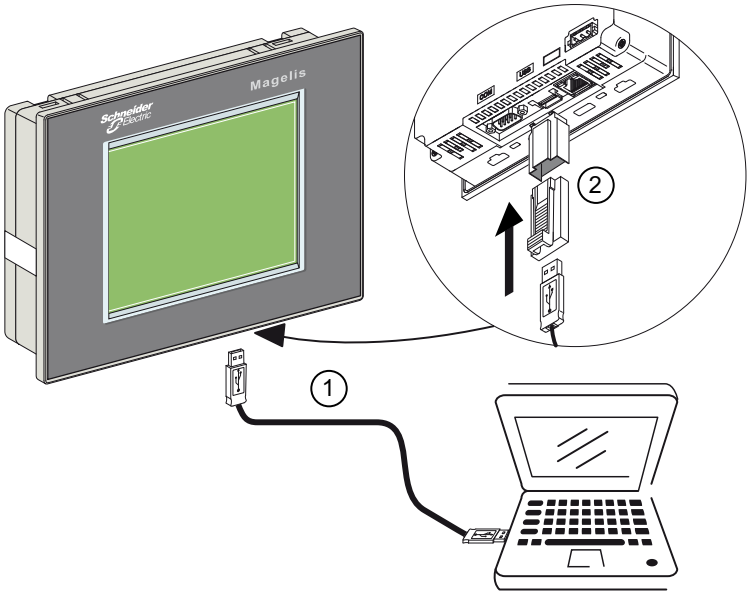
**NOTE:**
When mounted on a front panel, use the following cables combinations:
- XBTZG935 and XBTZGUSB
- TCSXCNAMUM3P and XBTZGUSBB

## Connecting with USB Cable

To connect the USB cable to your XBTGC HMI Controller, follow the steps in this table:

| Step | Action |
|---|---|
| 1 | Connect the USB cable to the XBTGC HMI Controller; check that the *USB holder (see Magelis XBTGC HMI Controller, Hardware Guide)* is in the correct position. |
| 2 | Connect your USB cable using the front panel *connections* *(see page 98)*. |
| 3 | Connect the USB cable to the PC. |

This diagram shows how to connect the XBTGC HMI Controller directly to a PC:

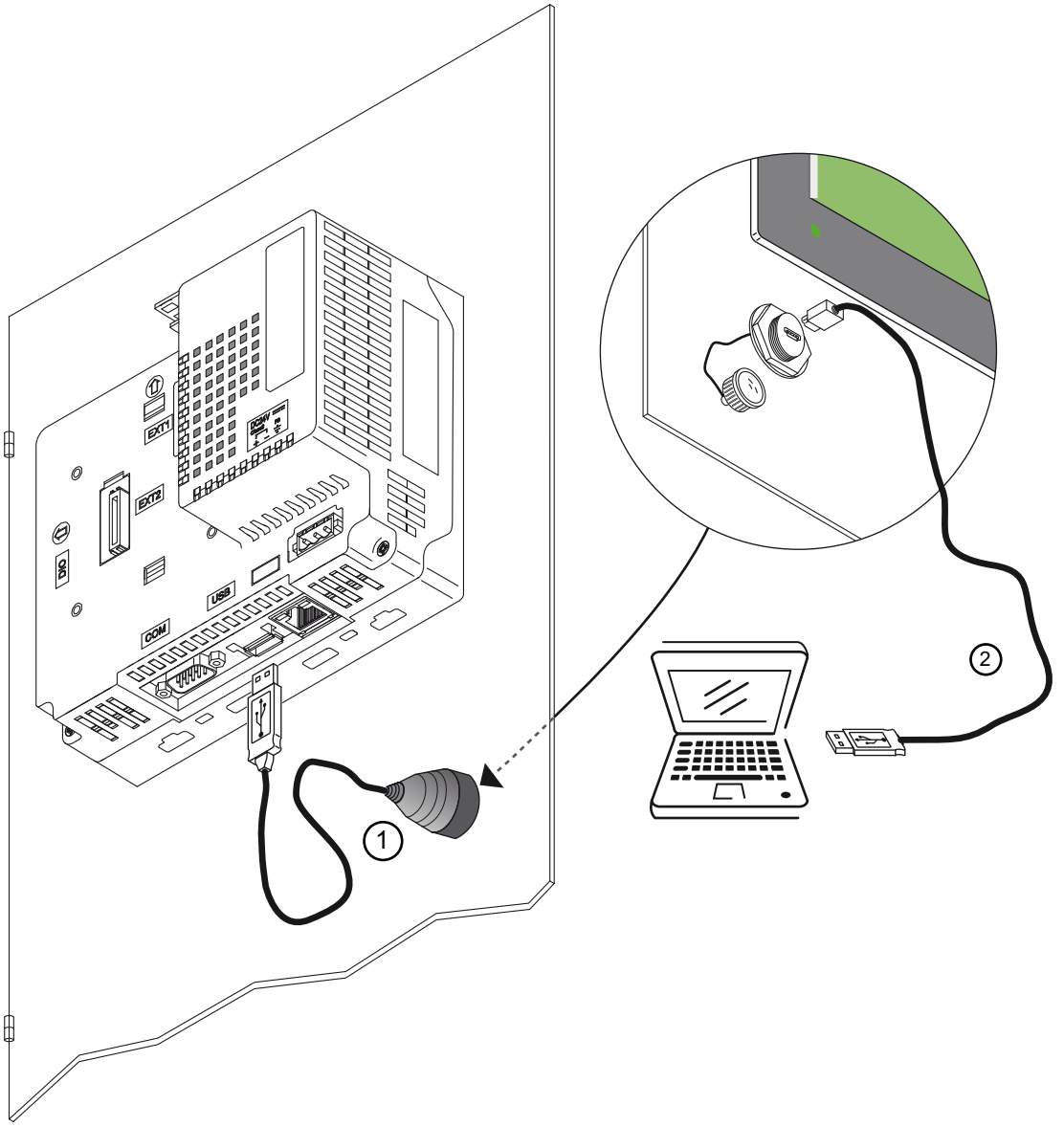



**Legend:**

**1:** USB data transfer cable (XBTZG935)

**2:** USB connection: refer to XBTGC HMI Controller *User Manual* for additional information on the USB holder.

This diagram shows how to connect the XBTGC HMI Controller to a PC, when mounted on a front panel:



**Legend:**

**1:** USB data transfer cable (XBTZGUSBB).

**2:** USB Min B to USB data transfer cable (TCSXCNAMUM3P or XBTZG935).

**NOTE:** An alternative download method consist of connecting your PC to any controllers via USB cable. Then connect your XBTGC HMI Controller to the first one via serial link. However transfer speed is slow.

## Application Download with Firmware Downgrade

The XBTGC HMI Controller 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.

| *NOTICE* |
| --- |
| **LOSS OF DATA**<br>Always save your application and firmware version on a USB memory key.<br>**Failure to follow these instructions can result in equipment damage.** |

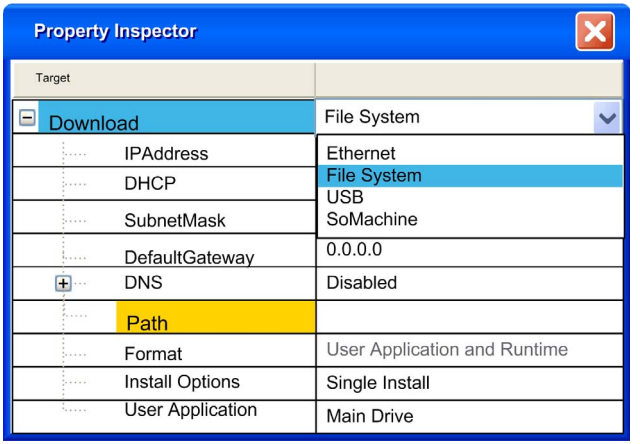
To download an application and downgrade the firmware of your controller, follow the steps in this table:

| Step | Action |
| --- | --- |
| 1 | Turn off the power supply of your controller, prior 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.<br>**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 an FAT32 USB memory key. To save, follow the steps in this table:

| Step | Action |
|---|---|
| 1 | Insert a USB memory key into the USB port of your computer. |
| 2 | Double-click **HMI Application** in the **Tools tree** tab of your project.<br>**Result:** The project switches for the HMI and the main Vijeo Designer window appears. |
| 3 | Right-click the controller node in the **Navigator** window, and select **Properties**.<br>**Result:** The **Property Inspector** window appears. |
| 4 | Select **File System** from the **Download** menu as shown in this figure:<br> |
| 5 | Set the directory from the **Path** menu to the USB memory key.<br>**NOTE:** Select the root level of your USB memory key. |
| 6 | Click the **OK** button.<br>**Result:** The directory is now set to the USB memory key. |
| 7 | Click **Build → Download All** from the Vijeo Designer main menu bar.<br>**Result:** The application is saved onto the USB memory key. |

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

# Chapter 15
# Troubleshooting and FAQ

## Introduction

This chapter contains common troubleshooting procedures and frequently asked questions for the XBTGC HMI Controller.

## What Is in This Chapter?

This chapter contains the following topics:


| Topic | Page |
|---|---|
| Troubleshooting | 104 |
| Frequently Asked Questions | 108 |

# Troubleshooting

## Introduction

This section lists the possible troubleshooting solutions with the XBTGC HMI Controller, 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:**

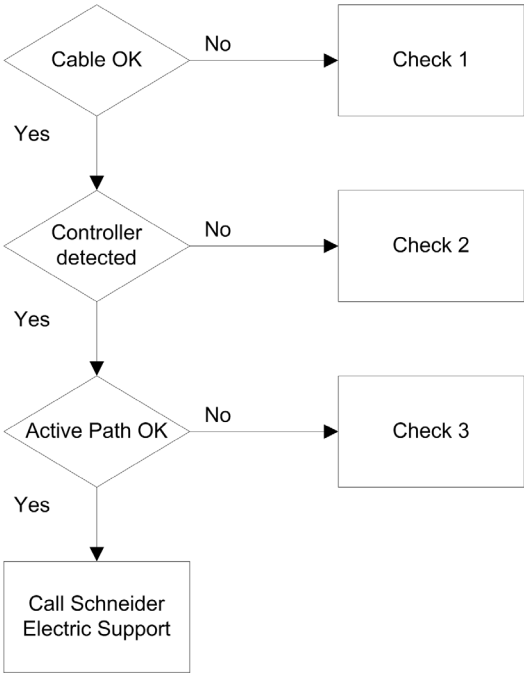
- Refer to Communication between SoMachine and the XBTGC HMI Controller *(see page 104)*.
- Your application program must be valid. Refer to the debugging section for more information.
- The CoDeSys gateway must be running:
  a. click the CoDeSys Gateway icon in the task bar,
  b. select **Start Gateway**.

## Communication Between SoMachine and the XBTGC HMI Controller is not Possible.

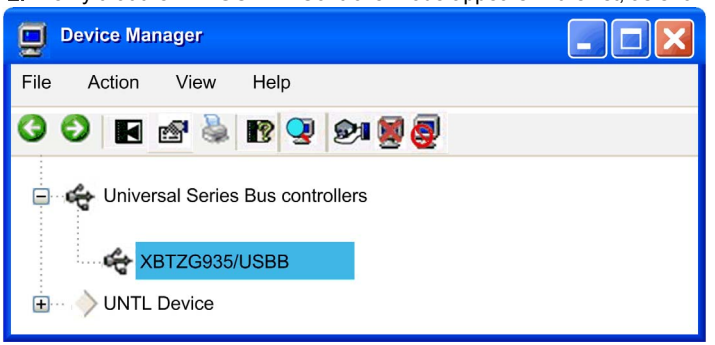
**Possible causes:**

- 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:<br>● The cable is correctly linked to the controller and to the PC, and is not damaged,<br>● You used the specific cable or adapter, depending on the connection type:<br>❍ Ethernet and Serial link connection.<br>❍ XBTZG935 cable for a USB connection.<br>❍ XBTZG935 and XBTZGUSB or TCSXCNAMUM3P and XBTZGUSBB connection when the controller is mounted on a front panel. |

| Check | Action |
|---|---|
| 2 | Verify that the XBTGC HMI Controller has been detected by your PC:<br>1. Click **Start → Control Panel → System**, then select the **Hardware** tab and click **Device Manager**,<br>2. Verify that the XBTGC HMI Controller node appears in the list, as shown below:<br><br>3. If the XBTGC HMI Controller 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 | Verify that the active path is correct:<br>1. Double-click the controller node in the device view.<br>2. Verify that the XBTGC HMI Controller node appears in bold, not in italic.<br>If not:<br>a. Stop the CoDeSys Gateway: right-click the icon in the task bar and select **Stop Gateway**.<br>b. Disconnect and reconnect the cable on the controller side.<br>c. Start the CoDeSys Gateway: right-click the icon in the task bar and select **Start Gateway**.<br>d. Select the gateway in the controller window of SoMachine and click **Scan network**. Select the XBTGC HMI Controller node and click **Set active path**.<br>**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 XBTGC HMI Controller node appears in italics. Select the XBTGC HMI Controller node and click **Resolve Name**. If the node no longer appears in italics, click **Set Active Path** to correct this. |

## 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 POUs.

### 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 XBTGC HMI Controller?

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 XBTGC HMI Controller Controller?

Refer to the *Supported Variables section* *(see page 24)*.

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

Communication is possible with an XBTGC HMI Controller only if the serial line is configured with the *Network Protocol* *(see page 92)*.

Limitations:

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

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

### When Should I Use Freewheeling or Cyclic Mode?

Freewheeling or cyclic mode usage:

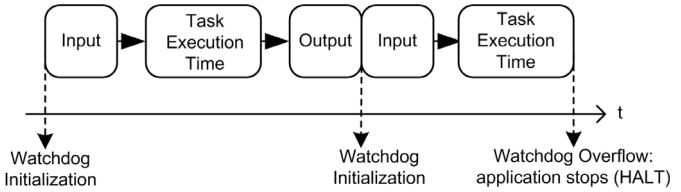
- Freewheeling: use this mode if you want a variable cycle time. The next cycle starts after a waiting period equal to 30% of the last cycle execution time.
- Cyclic: use this mode if you want to control the frequency cycle.

## 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.

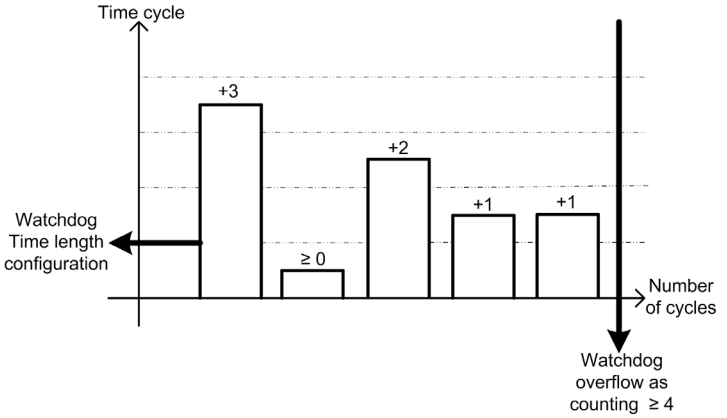
Depending on the **Time** and **Sensitivity** parameters, if the watchdog is triggered, the controller is stopped and goes into HALT mode. The associated task remains uncompleted, as shown in this diagram:




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 application 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 XBTGC HMI Controller through several USB ports on my PC?

No, this is not supported.

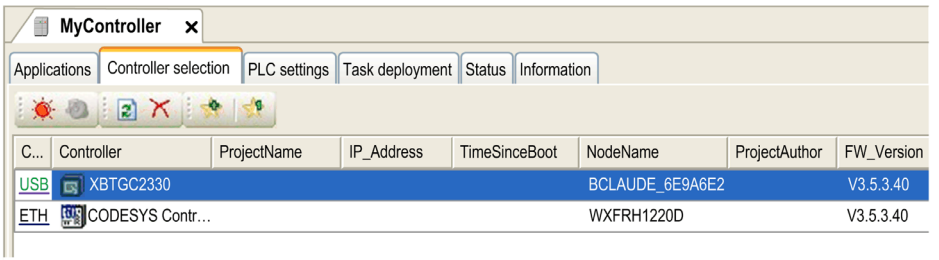
### When I use a new controller in SoMachine application with a previously used HMI application, why do the 2 applications no longer communicate?

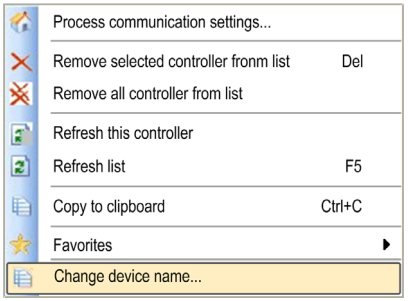
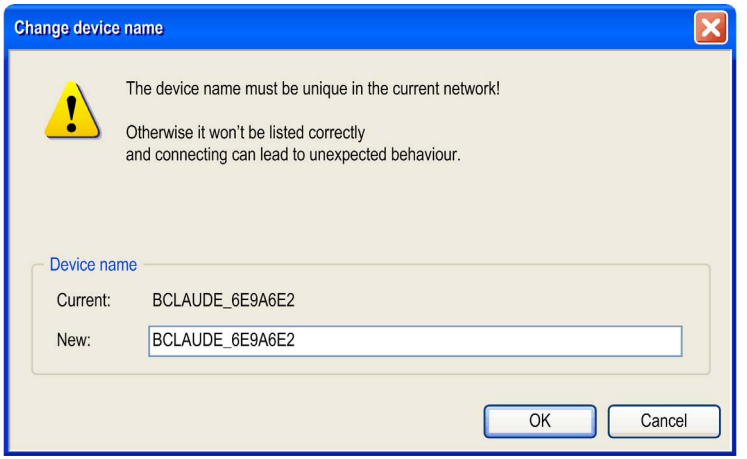
This is because the controller name in the HMI application (Vijeo-Designer) is not updated. The HMI application is configured with the previous controller name; it is necessary update this application with the SoMachine controller name.

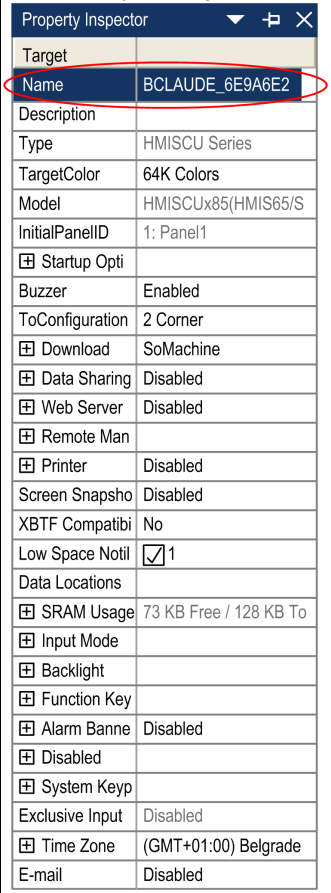
The following procedure updates the HMI application controller name with the SoMachine controller name. However, you may update the SoMachine controller name with the HMI application controller name, refer to update controller name using the HMI application *(see page 113)*.

### 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:

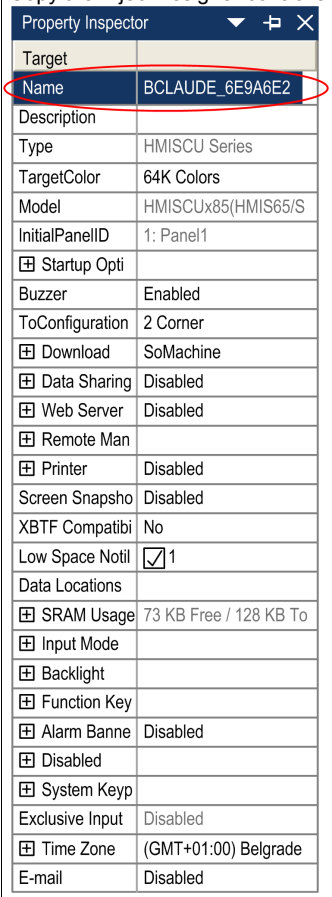
| Step | Action |
|---|---|
| 1 | Display the **SoMachine Logic Builder**. |
| 2 | Double-click the controller in the **Devices tree**.<br>**Result:** The device editor window opens. |
| 3 | Select the **Controller selection** tab.<br>**Result**: The **Controller selection** tab opens:<br> |

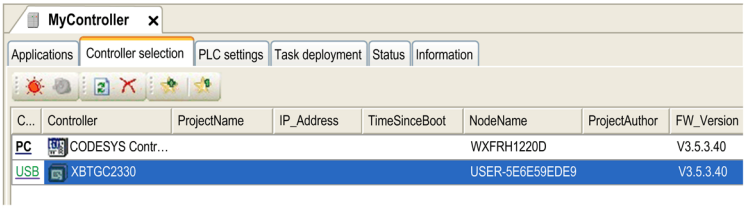
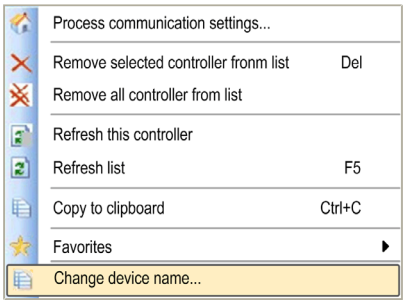
| Step | Action |
|---|---|
| 4 | Right-click the controller.<br>**Result:** The controller contextual menu opens.<br> |
| 5 | Select `Change device name...`.<br>**Result:** The **Change device name** dialog opens:<br> |
| 6 | Make sure that device name meets the Vijeo-Designer controller name requirements: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

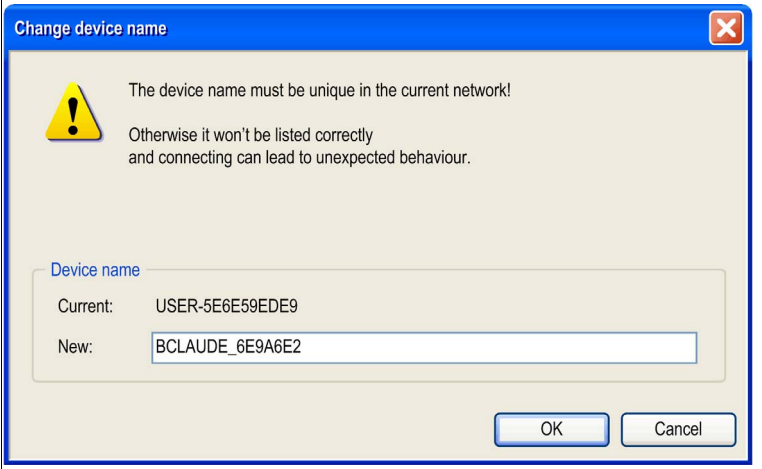
| Step | Action |
|---|---|
| 10 | Paste the Vijeo-Designer controller name in the **Property Inspector → Name** field:  |
| 11 | Press **Enter** to apply the change to the controller name. |

## 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 | Copy the Vijeo-Designer controller name from the **Property Inspector → Name** field:<br><br> |
| 3 | Display the **SoMachine Logic Builder**. |
| 4 | Double-click the controller in the **Devices tree**.<br>**Result:** The device editor window opens. |

| Step | Action |
|---|---|
| 5 | Select the **Controller selection** tab.<br>**Result**: The **Controller selection** tab opens:<br> |
| 6 | Right-click the controller.<br>**Result:** The controller contextual menu opens.<br> |
| 7 | Select `Change device name...`.<br>**Result:** The **Change device name** dialog opens:<br> |

| Step | Action |
|---|---|
| 8 | Paste the controller name into the **New** field.<br> |
| 9 | Click **OK** to apply the change to the controller name. |

## How do I select the XBTGC HMI Controller boot up behavior (RUN or STOP) after a power cycle?

The RUN/STOP state of the XBTGC HMI Controller 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 XBTGC HMI Controller 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 XBTGC HMI Controller 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 µs) to run, the Task Monitor will show 0 µs.

# Glossary

## !

**%**

According to the IEC standard, % is a prefix that identifies internal memory addresses in the logic controller to store the value of program variables, constants, I/O, and so on.

**%I**

According to the IEC standard, %I represents an input bit (for example, a language object of type digital IN).

**%MW**

According to the IEC standard, %MW represents a memory word register (for example, a language object of type memory word).

**%Q**

According to the IEC standard, %Q represents an output bit (for example, a language object of type digital OUT).

## A

**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.

**ARRAY**

The systematic arrangement of data objects of a single type in the form of a table defined in logic controller memory. The syntax is as follows: `ARRAY [<dimension>] OF <Type>`

Example 1: `ARRAY [1..2] OF BOOL` is a 1-dimensional table with 2 elements of type `BOOL`.

Example 2: `ARRAY [1..10, 1..20] OF INT` is a 2-dimensional table with 10 x 20 elements of type `INT`.

**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.

# 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.

**CANopen**

An open industry-standard communication protocol and device profile specification (EN 50325-4).

**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.

**cyclic task**

The cyclic scan time has a fixed duration (interval) specified by the user. If the current scan time is shorter than the cyclic scan time, the controller waits until the cyclic scan time has elapsed before starting a new scan.

## D

**DHCP**

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

**digital I/O**

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

**DINT**

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

**DNS**

(*domain name system*) The naming system for computers and devices connected to a LAN or the Internet.

**DWORD**

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

## E

**EDS**

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

**element**

The short name of the ARRAY element.

**encoder**

A device for length or angular measurement (linear or rotary encoders).

**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.

## F

**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.

**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**

A programming unit that has 1 input and returns 1 immediate result. However, unlike FBs, it is directly called with its name (as opposed to through an instance), has no persistent state from one call to the next and can be used as an operand in other programming expressions.

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

## H

**HMI**

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

**HSC**

(*high-speed counter*)

## I

**I/O**

(*input/output*)

**IL**

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

**input filter**

A special function that helps reject extraneous signals on input lines due to such things as contact bounce and inducted electrical transients. Inputs provide a level of input filtering using the hardware. Additional filtering with software is also configurable through the programing or the configuration software.

**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

**LCD**

(*liquid crystal display* Used in many HMI devices to display menus and messages to machine operators.

**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`).

**located variable**

Refer to (*unlocated variable*).

**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.

**master/slave**

The single direction of control in a network that implements the master/slave mode.

**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.

**node**

An addressable device on a communication network.

# O

**OS**

(*operating system*) A collection of software that manages computer hardware resources and provides common services for computer programs.

# 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. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs 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.

**protocol**

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

**PTO**

(*pulse train outputs*) a fast output that oscillates between off and on in a fixed 50-50 duty cycle, producing a square wave form. 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.

**RTU**

(*remote terminal unit* ) A device that interfaces with objects in the physical world to a distributed control system or SCADA system by transmitting telemetry data to the system and/or altering the state of connected objects based on control messages received from the system.

# S

**scan**

A function that includes:
- reading inputs and placing the values in memory
- executing the application program 1 instruction at a time and storing the results in memory
- using the results to update outputs

**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.

**STN**

(*super-twisted nematic* A display technology (type of monochrome passive-matrix liquid crystal display).

**STOP**

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

**string**

A variable that is a series of ASCII characters.

**symbol**

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

**system variable**

A variable that provides controller data and diagnostic information and allows sending commands to the controller.

## T

**task**

A group of sections and subroutines, executed cyclically or periodically for the master task, or periodically for the periodic task.

A task possesses a priority level and is linked to the I/Os of the logic controller. These I/Os are refreshed in consequence.

A logic controller can have several tasks.

**TFT**

(*thin film transmission*) A technology used in many HMI display devices (also known as active matrix).

**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

**watchdog**

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

**WORD**

A type encoded in a 16-bit format.

# Index


## A

Adding
- Expansion Modules, *19*
- CANopen Module, *18*
- Controller, *17*
- Devices, *16*
- Expansion Module, *19*

Addressing Modes differences, *29*
Analog I/O Modules
- TM2, *82*

Application
- Active, *14*
- Save, *102*

Array
- Data Exchange, *25*

## C

CANopen
- Adding Module, *18*, *18*
- Baudrate Configuration, *87*
- Expansion Modules, *89*
- Hardware Configuration, *86*
- Interface Configuration, *86*
- Master Unit, *86*
- Network Manager, *87*
- Optimized Manager, *88*
- Remote Devices, *89*, *89*
- Remote Devices Configuration Screen, *89*
- Software Requirements, *86*

Characteristics
- Controller, *13*

Combination
- Special I/O, *72*

Configuration
- Baudrate Configuration for CANopen, *87*
- CANopen, *85*
- CANopen Hardware Configuration, *86*
- CANopen Interface, *86*
- CANopen Software Requirements, *86*
- Controller Hardware Configuration, *19*
- Embedded I/O, *65*
- Embedded I/O Configuration Editor, *65*
- Ethernet, *83*, *83*
- I/O Expansion Modules, *79*
- IP Address Configuration, *83*
- Optimized Manager, *88*
- Serial Line, *91*
- Special I/O, *69*

Controller
- Adding, *17*
- Characteristics, *13*
- Connecting the controller, *97*
- Creating Projects, *12*
- Hardware Configuration, *19*
- Libraries, *21*
- Memory, *27*, *28*
- Tasks, *31*

Controller Configuration
- Controller, *63*

Creating
- New Project, *13*
- Projects, *12*, *13*

## D

Data Exchange
- Array, *25*
- Structure, *25*

Device Editor
- Controller Device Editor, *63*
- Tabs, *64*

Devices
- Adding, *16*
- Tree, *15*

Devices Editor
Window, *63*
Digital I/O Modules
TM2, *81*
Download
Application, *101*
USB, *98*
Download application, *59*

## E

Editor
Controller Device Editor, *63*
Embedded I/O Configuration, *65*
Embedded I/O Configuration
Editor, *65*
I/O Mapping Tabs, *66*
I/O Mapping Tabs Parameters, *67*
Tabs, *67*
Tabs Parameters, *68*
Ethernet
Configuration, *83*, *83*
Exchange
Variables, *26*
Expansion Modules
Adding, *19*
CANopen, *86*, *89*
Considerations, *80*
I/O Expansion Modules, *79*
Maximum Hardware Configuration, *80*

## F

FAQ, *108*
Connecting Multiple Controller through
USB Ports, *110*
Controller and HMI Communication, *110*
Controller Boot State, *115*
Controller Name Update, *110*, *113*
SoMachine Network Communication, *108*
Start all application checkbox, *110*
Supported Programming Languages, *108*
Supported Variables, *108*
Task Mode, *108*
Task Monitor, *115*
Watchdog Configuration, *109*
Firmware
Downgrade, *101*
Save, *102*
Update, *97*

## I

I/O
Digital I/O, *81*
Embedded I/O, *65*
Expansion Modules, *79*
I/Os
Summary, *76*
IP Address
Configuration, *83*
Default, *84*

## L

Libraries
Controller, *21*
Local and Special I/O
Overview, *70*

## M

Memory
Controller, *27*
Mapping, *28*
Modbus Manager, *95*

## N

Network Manager
CANopen, *87*
Serial Line, *93*

## O

Overview
Local and Special I/O, *70*

## P

Project
Creating New Project, *13*

## R

Reboot, *57*
Transfer, *97*
Remanent variables, *62*
Reset cold, *56*
Reset origin, *57*
Reset warm, *56*
Run command, *55*

## S

Save
Application, *102*
Firmware, *102*
USB, *102*
Serial Line
Configuration, *91*, *92*
Configuration Window, *92*
Modbus Manager, *95*
Network Manager, *93*
Serial Link
SoMachine Network Manager, *94*
SoMachine Network Manager, *94*
Special I/O
Combination, *72*
Special I/O Configuration
Configuration, *69*
State diagram, *44*
Stop command, *55*
Structure
Data Exchange, *25*
Summary
I/Os, *76*
Supported Standard Data Types
Supported Variables, *23*
Supported Variables
Types, *24*

## T

Task
Controller tasks, *31*
Cyclic task, *36*
Event task, *37*
Freewheeling task, *37*
Types, *36*
Watchdogs, *38*
Troubleshooting, *104*
Application Transfer, *104*
Boot Application, *107*
CANopen Heartbeat, *107*
Communication, *104*
Device Name, *107*
Out of Memory, *107*
POU Monitoring, *107*
RUN State, *106*

## U

USB
Connection, *98*
Save, *102*

## V

Variables
Exchange, *26*