Modicon TM3 Expert I/O Modules

HSC Library Guide

05/2019

EIO0000003683.00

www.schneider-electric.com

Schneider Electric

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

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

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

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

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

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

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

This document describes how to use the high speed counter (HSC) library of the TM3 expert modules.

To use this manual, you must:

- Have a thorough understanding of the TM3 expert modules, including their design, functionality, and implementation within control systems.
- Be proficient in the use of the following IEC 61131-3 PLC programming languages:
  - Function Block Diagram (FBD)
  - Ladder Diagram (LD)
  - Structured Text (ST)
  - Instruction List (IL)
  - Sequential Function Chart (SFC)

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.1.

Related Documents

Title of Documentation	Reference Number
EcoStruxure Machine Expert - Programming Guide	*EIO0000002854 (ENG)* *EIO0000002855 (FRE)* *EIO0000002856 (GER)* *EIO0000002857 (SPA)* *EIO0000002858 (ITA)* *EIO0000002859 (CHS)*
Modicon TM3 Expansion Modules Configuration - Programming Guide (EcoStruxure Machine Expert)	*EIO0000003119 (ENG)* *EIO0000003120 (FRA)* *EIO0000003121 (GER)* *EIO0000003122 (SPA)* *EIO0000003123 (ITA)* *EIO0000003124 (CHS)*
TM3 Expert Modules - Hardware Guide	*EIO0000003137 (ENG)* *EIO0000003138 (FRE)* *EIO0000003139 (GER)* *EIO0000003140 (SPA)* *EIO0000003141 (ITA)* *EIO0000003142 (CHS)* *EIO0000003428 (POR)* *EIO0000003429 (TUR)*

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

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

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

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

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

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

Part I
Introduction

Overview

This part provides an overview description, available modes, functionality and performances of the different functions.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Expert Function Introduction	17
2	High Speed Counter Types	21

Chapter 1
Expert Function Introduction

Expert Functions Overview

Introduction

The inputs and outputs available on TM3 HSC expansion modules can be configured to use High Speed Counter (HSC) expert functions.

I/O characteristics:

I/O Type	TM3XFHSC202	TM3XHSC202
Fast inputs	10	
Fast outputs	8	
External Events	8	None

The HSC functions execute fast counts of pulses from sensors, switches, or other equipment connected to the fast inputs.

For more information about the HSC functions, refer to High Speed Counter Types *(see page 21)*.

Maximum Number of Expert Functions

The maximum number of expert functions that can be configured depends on the expert function type and number of optional functions *(see page 123)* configured. Refer to I/O Assignment *(see page 19)*. You can configure up to two HSC Main, and if you do not configure all the options, you can configure HSC Simple counters for those inputs that remain.

Maximum number of expert functions by TM3 reference:

Expert Function Type		TM3XFHSC202 TM3XHSC202
Total number of HSC functions		10
HSC Simple		10
HSC Main	Single Phase	2
	Dual Phase	
	Frequency Meter	
	Period Meter	

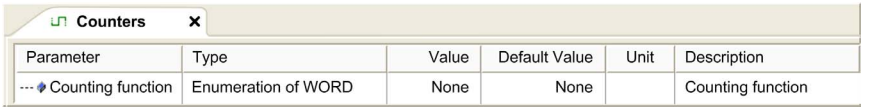
The maximum number of expert functions possible is further limited by the number of inputs used by each expert function.

Example configurations:

- 10 HSC Simple on TM3XHSC202
- 2 HSC Main Single Phase with EN, CAP and SYNC configured + 2 HSC Simple

Configuring an Expert Function

To configure an expert function, proceed as follows:

Step	Description
1	Add a TM3 expert module to your configuration.
2	In the **Devices tree**, double-click **MyController** → **IO_Bus** → **Module_x** → **Counters**, where x is the module number. **Result:** The **Counters** configuration window appears: 
3	Double-click **None** in the **Value** column and choose the counter type to assign. **Result:** The default configuration of the counter type appears when you click anywhere in the configuration window.
4	Configure the expert function parameters, as described in the following chapters.
5	To configure an additional expert function, click the **+** tab that appears when you have configured an expert function. **NOTE:** If the maximum number of expert functions is already configured, a message appears at the bottom of the configuration window informing you that you can now add only **HSC Simple** functions.

I/O Configured as Expert Functions

When I/Os are configured as expert functions:

- I/O can be read through memory variables.
- Short-circuit management applies on the outputs. Status of outputs are available.
- When inputs are used in HSC expert functions, the integrator filter is replaced by an anti-bounce filter. The filter value is configured in the configuration screen.

I/Os that are not used by expert functions can be configured as regular I/Os.

I/O Assignment

The following I/Os can be configured for use by expert functions:

	TM3XFHSC202 TM3XHSC202
Inputs	10 fast inputs (I0...I9)
Outputs	8 fast outputs (Q0...Q7)

NOTE: All I/Os are by default disabled in the configuration window.

The following table shows the I/Os that can be configured for expert functions:

Expert Function	Name	Fast Input	Fast Output
HSC Simple	A input	M	–
HSC Main Single Phase	A input	M	–
HSC Main Dual Phase	A input B input	M M	– –
Frequency Meter	A Input	M	–
Period Meter	A Input	M	–
Optional Functions	EN input	C	–
	SYNC input	C	–
	CAP input	C	–
	Reflex 0 output	–	C
	Reflex 1 output	–	C
	Reflex 2 output	–	C
	Reflex 3 output	–	C
M Mandatory **C** Optionally configurable			

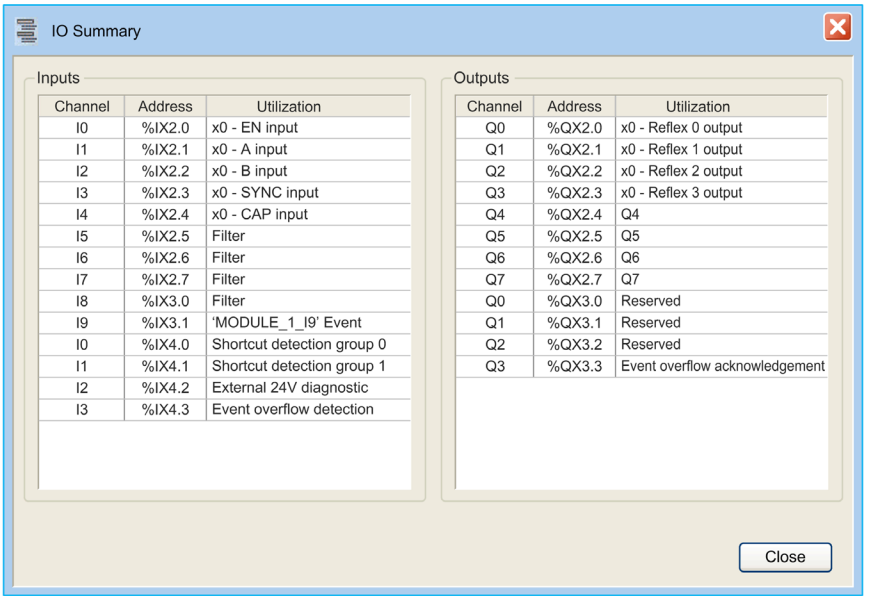
I/O Summary

The **IO Summary** window displays the I/Os and external events being used by the expert functions or the expansion modules.

To display the **IO Summary** window:

Step	Action
1	In the **Devices tree** tab, right-click the **Module_x** node and choose **IO Summary**.

Example of **IO Summary** window:



Chapter 2
High Speed Counter Types

Overview

This chapter provides an overview of the different types of HSC.

What Is in This Chapter?

This chapter contains the following topics:

Choosing Your Counter

Overview

Start the HSC configuration by choosing a counter type according to the application need.

In the **Counters** editor, select a **Counting function** from the list that offers the following types of counters:

- **HSC Simple**
- **HSC Main Single Phase**
- **HSC Main Dual Phase**
- **Frequency Meter**
- **Period Meter**

The **Frequency Meter** type and the **Period Meter** type are both based on an **HSC Main** type.

For each counter defined in the **Counters** editor, a default **Instance name** is assigned by EcoStruxure Machine Expert. This default **Instance name** is editable. You must use the instance name as an input to the function blocks affecting the counter.

Type and Mode Matrix

This table presents the different types and modes available:

Type Mode	HSC Simple	HSC Main Single Phase	HSC Main Dual Phase	Frequency Meter	Period Meter
One-shot	X	X	–	–	–
Modulo-loop	X	X	X	–	–
Event Counting	–	X	–	–	–
Free-large	–	–	X	–	–
Edge to Edge	–	–	–	–	X
Edge to Opposite	–	–	–	–	X

HSC Simple

This table presents an overview of the specifications available in **HSC Simple** type according to the mode requested:

Feature	Function	
	One-shot Mode	Modulo-loop Mode
Counting mode	Count down	Count up
Enable with an HSC physical input	No	No
Synchronization / preset with an HSC physical input	No	No
Comparison function	No	No
Capture function	No	No
Trigger external event on stop	Yes(1)	No
(1) TM3XFHSC202 modules only		

HSC Main Single Phase

This table presents an overview of the specifications available in **HSC Main Single Phase** type according to the mode requested:

Feature	Function		
	One-shot Mode	Modulo-loop Mode	Event Counting Mode
Counting mode	Count down	Count up	Pulse counting during given time base (10 ms, 100 ms, or 1000 ms)
Enable with an HSC physical input	Yes	Yes	No
Synchronization / preset with an HSC physical input	Yes	Yes	Yes
Comparison function	Yes, 4 thresholds, 4 reflex outputs, and 4 external events(1)	Yes, 4 thresholds, 4 reflex outputs, and 4 external events(1)	No
Capture function	Yes, 1 capture register	Yes, 1 capture register	No
Trigger external event on stop	Yes(1)	No	No
(1) TM3XFHSC202 modules only			

HSC Main Dual Phase

This table presents an overview of the specifications available in **HSC Main Dual Phase** type according to the mode requested:

Feature	Function
Counting mode	Count up / down Pulse / direction Quadrature
Enable with an HSC physical input	Yes
Synchronization / preset with an HSC physical input	Yes
Comparison function	Yes, 4 thresholds, 4 reflex outputs, and 4 external events(1)
Capture function	Yes, 1 capture register
(1) External events are supported by TM3XFHSC202 modules only.	

Frequency Meter

This table presents an overview of the specifications available in **Frequency Meter** type:

Feature	Function
Counting mode	Pulse frequency in Hz with updated value available every time base value (10 ms, 100 ms, or 1000 ms)
Enable with an HSC physical input	Yes
Synchronization / preset with an HSC physical input	No
Comparison function	No
Capture function	No

Period Meter

This table presents an overview of the specifications available in **Period Meter** type according to the mode requested:

Feature	Function
Counting modes	Edge to edge: Measure the time between two events Edge to opposite: Measure the duration of an event
Enable with an HSC physical input	Yes
Synchronization / preset with an HSC physical input	No
Comparison function	Yes, 4 thresholds, 4 reflex outputs, and 4 external events(1)
Capture function	No
Resolution	Duration counting with configurable resolution (0.1 µs, 1 µs, 100 µs, or 1000 µs)
Timeout	0...858993459, calculated using resolution units 0 means no timeout
Trigger external event at end of period	Yes(1)
(1) External events are supported by TM3XFHSC202 modules only.	

Simple Type Overview

Overview

The **Simple** type is a single input counter.

Any operation on the counter (enable, sync) and any action triggered (when count value is reached) is executed in the context of a task.

With the **Simple** type and the TM3XFHSC202 module, you can trigger a stop event.

Simple Type Modes

The **Simple** type supports 2 configurable counting modes on single-phase pulses:

One-shot *(see page 35)*. In this mode, the counter value register decrements (from a user-defined value) for each pulse applied to A input, until the counter reaches 0.

Modulo-loop *(see page 55)*. In this mode, the counter repeatedly counts from 0 to a user-defined modulo value then returns to 0 and restarts counting.

Performance

The maximum frequency admissible on a fast input is 100 kHz if the bounce filter value is 0.005 ms. If the bounce filter value is 0.002 ms (default value for configuration), the maximum frequency is 200 kHz.

For more information about the bounce filter, refer to Dedicated Features *(see page 148)*.

Main Type Overview

Overview

The **Main** type is a counter that uses up to 5 fast inputs and 4 reflex outputs. The TM3 HSC expansion modules can have up to 2 **Main** type high speed counters configured.

Main Type Modes

The **Main** type supports the following counting modes on single phase (1 input) or dual-phase (2 inputs) pulses:

One-shot *(see page 41)*: In this mode, the counter value register decrements (from a user-defined value) for each pulse applied to the A input until the counter reaches 0.

Modulo-loop *(see page 61)*: In this mode, the counter repeatedly counts up from 0 to a user-defined modulo value, then returns to 0 and restarts counting. In reverse, the counter counts down from the modulo value to 0, then presets to the modulo value and restarts counting.

Free-large *(see page 79)*: In this mode, the counter behaves like a full range up and down counter.

Event Counting *(see page 91)*: In this mode, the counter accumulates the number of events that are received during a user-configured time base.

Frequency meter *(see page 101)*: In this mode, the counter measures the frequency of events. Frequency is the number of events per second (Hz).

Period meter *(see page 113)*: Use the **Period meter** mode to:
- determine the duration of an event
- determine the time between 2 events
- set and measure the execution time for a process

Optional Features

Optional features can be configured depending on the selected mode:
- Hardware inputs to operate the counter (enable, preset) or capture the on-going counting value
- Up to 4 thresholds, the values of which can be compared.
- With the TM3XFHSC202 module, up to 4 external events (1 for each threshold) can be associated with tasks
- Up to 4 reflex outputs

Performance

The maximum frequency admissible on a fast input is 100 kHz if the bounce filter value is 0.005 ms. If the bounce filter value is 0.002 ms (default value for configuration), the maximum frequency is 200 kHz.

For more information about the bounce filter, refer to Dedicated Features *(see page 148)*.

Frequency Meter Type Overview

Overview

The **Frequency Meter** type is a counter that uses up to 2 fast inputs. Each TM3 HSC expansion module can have up to 2 **Frequency Meter** type high speed counters configured.

Frequency Meter Type Mode

The **Frequency meter** *(see page 101)* counter measures the frequency of events. Frequency is the number of events per second (Hz).

Performance

The maximum counter frequency is determined by the **Filter** value, used to reduce the bounce effect on the input. The maximum frequency admissible on a fast input is 200 kHz if the bounce filter value is 0.002 ms (the default value).

Configure the bounce filter value according to the frequency applied at the inputs. If the filter value is too high, the input frequency may not be detected and the counter value (`CurrentValue` output) of the HSCMain_TM3 *(see page 164)* function block is set to 0.

Refer also to TM3XFHSC202 Input Characteristics *(see Modicon TM3, Expert I/O Modules, Hardware Guide)* and TM3XHSC202 Input Characteristics *(see Modicon TM3, Expert I/O Modules, Hardware Guide)*.

For more information about the bounce filter, refer to Dedicated Features *(see page 148)*.

Accuracy

The precision of the Frequency meter counter cannot be assumed when the measured frequency falls below a specific minimum value. This minimum value depends on the time base configured, that is, the period of time during which events are counted.

This table shows for each configurable value of **Time base** the minimum frequency necessary to obtain an accuracy of 0.01 %:

Configured Time Base (ms)	Minimum Value (Hz)
10	>200
100	>20
1000 (default)	>2

The maximum duty cycle at 200 kHz is 55 %.

Period Meter Type Overview

Overview

The **Period Meter** type is a counter that uses up to 2 fast inputs.

Each TM3 HSC expansion module can have up to 2 **Period Meter** type high speed counters configured.

Period Meter Type Mode

Use the **Period meter** counting mode to:

- Determine the duration of an event
- Measure the time between 2 events
- Set and measure the execution time for a process

Performance

Performance is determined by the **Filter** value, used to reduce the bounce effect on the input.

For more information about the bounce filter, refer to Dedicated Features *(see page 148)*.

Accuracy

A maximum error applies to each measurement.

This table shows the maximum error for each configurable value of **Resolution**:

Configured Resolution (µs)	Maximum Error (µs)
0.1	0.5
1 (default	0.5
100	3
1000	3

For example, if the configured resolution is 0.1 µs, a generated pulse of 1 µs is displayed between 5 (0.5 µs) and 15 (1.5 µs).

Part II
One-shot Mode

Overview

This part describes the use of a HSC in **One-shot** Mode.

What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
3	**One-shot** Mode Principle	33
4	**One-shot** with a **Simple** Type	35
5	**One-shot** with a **Main** Type	41

Chapter 3
One-shot Mode Principle

One-shot Mode Principle Description

Overview

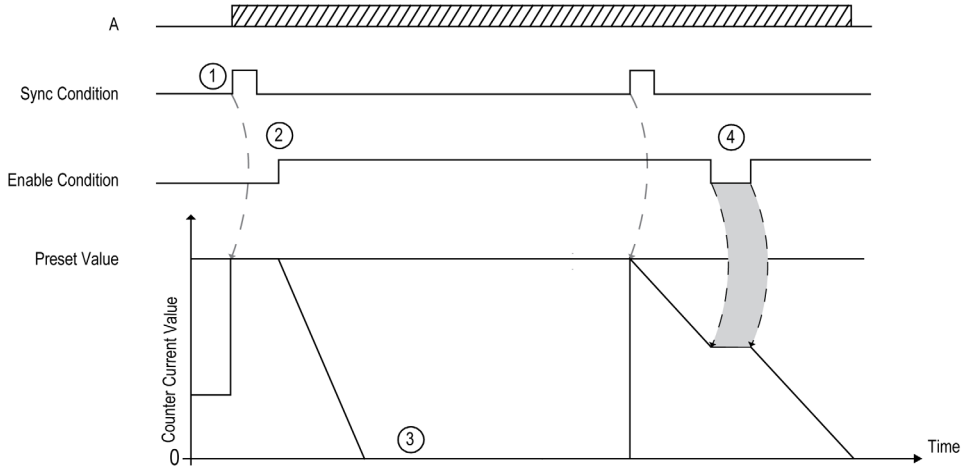
The counter is activated by a synchronization edge, and the preset value is loaded.

When counting is enabled, each pulse applied to the input decrements the value. The counter stops when its value reaches 0.

The counter value remains at 0 even if new pulses are applied to the input.

A new synchronization is needed to activate the counter again.

Principle Diagram



This table explains the stages from the preceding graphic:

Stage	Action
1	On the rising edge of the Sync condition, the preset value is loaded in the counter (regardless of the value of the counter at the time) and the counter is initialized.
2	When the Enable condition = 1, the counter value decrements on each pulse on input A until it reaches 0.
3	The counter waits until the next rising edge of the Sync condition. **Note:** At this point, pulses on input A have no effect on the counter.
4	When the Enable condition = 0, the counter ignores the pulses from input A and retains its value until the Enable condition again = 1. The counter resumes counting pulses from input A on the rising edge of the Enable input from the held value.

NOTE: Enable and Sync conditions depends on configuration. These are described in the Enable *(see page 128)* and Preset *(see page 126)* function.

Chapter 4
One-shot with a Simple Type

Overview

This chapter describes how to implement a High Speed Counter in **One-shot** mode using a **Simple** type.

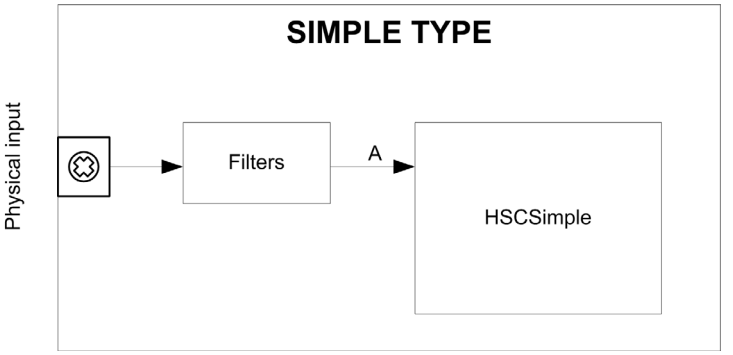
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Simple** type in **One-shot** mode:



A is the counting input of the High Speed Counter. **Simple** type counting for **One-shot** mode always counts down.

Configuration of the Simple Type in One-Shot Mode

Procedure

Follow this procedure to configure a **Simple** type in **One-shot** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: The **Counters** editor tab opens for HSC configuration.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Simple**, then click anywhere in the configuration area. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, modify the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **One-shot**.
5	In **Counting inputs → A input → Location** select the input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
7	Enter the value of the **Range → Preset** parameter to set the counting initial value.
8	With a TM3XFHSC202 expansion module, you can specify the name of an external event *(see page 143)*. When this event is triggered in a task, the counter is stopped. Set the value of **Stop → Stop event** to **Yes**, then modify the **Stop Event Name** to the name of the external event.

Programming the Simple Type

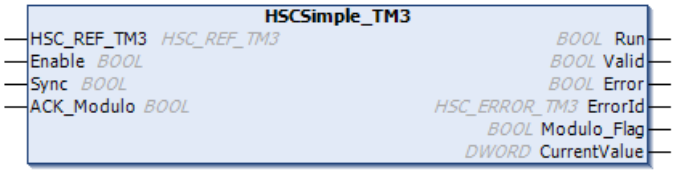
Overview

A **Simple** type counter is always managed by an `HSCSimple_TM3` *(see page 162)* function block.

NOTE: At build time, an error is detected if the `HSCSimple_TM3` function block is used to manage a different HSC type.

Adding an HSCSimple Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3_HSC → HSC → HSCSimple_TM3** in the list.
2	Drag-and-drop the item onto the **POU** window.
3	Edit the default **Simple** type instance name to match the **Instance name** of the counter function block defined in the Configuration window.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **One-shot** mode.

This table describes the input variables:

Input	Type	Comment
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC instance.
Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
Sync	BOOL	On rising edge, loads the preset of the counter.
ACK_Modulo	BOOL	Not used in one-shot mode.

This table describes the output variables:

Output	Type	Comment
Run	BOOL	Set to 1 when the counter is activated. Set to FALSE when the counter value reaches 0. A synchronization is needed to restart the counter.
Valid	BOOL	TRUE = indicates that output values on the function block are valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See the HSC_ERROR_TM3 *(see page 152)* enumeration.
Modulo_Flag	BOOL	Not used in one-shot mode.
CurrentValue	DWORD	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the HSCGetParam_TM3 *(see page 171)* or HSCSetParam_TM3 *(see page 169)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M262 Logic/Motion Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the HSC_PARAMETER_TYPE_TM3 *(see page 155)* that can be read or modified while the program is running:

Parameter	Description
HSC_PRESET	to get or set the Preset value of an HSC

Chapter 5
One-shot with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **One-shot** mode using a **Main** type.

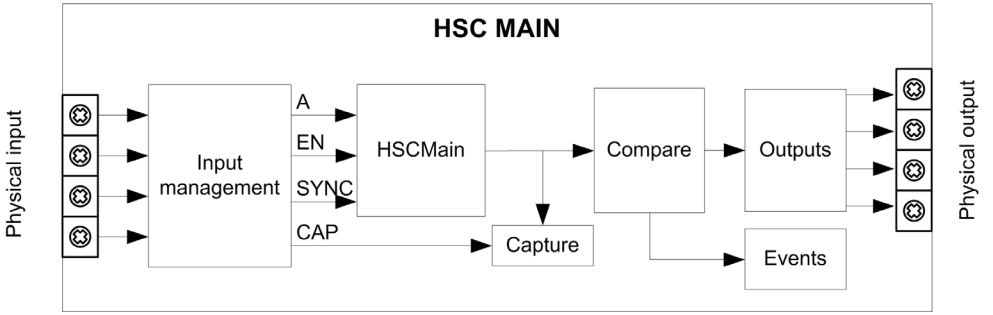
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **One-shot** mode:



A is the counting input of the counter.

EN is the enable input of the counter.

SYNC is the synchronization input of the counter.

CAP is the capture input of the counter.

Optional Function

In addition to the **One-shot** mode, relative to the **Simple** type, the **Main** type can provide the following functions:

- Preset function *(see page 126)*
- Enable function *(see page 128)*
- Capture function *(see page 131)*
- Comparison function *(see page 135)*

Configuration of the Main Type Single Phase in One-Shot Mode

Procedure

Follow this procedure to configure a **Main** type single phase in **One-shot** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: The **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Single Phase** and click anywhere in the configuration window. **Result**: The configuration parameters appear in the **Counters** tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **One-shot**.
5	In **Counting Inputs → A input → Location** select the input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
7	Enter the value of the **Range → Preset** parameter to set the initial counting value of the Preset function *(see page 126)*.
8	Optionally, you can enable these functions: • Preset function *(see page 126)* • Enable function *(see page 128)* • Capture function *(see page 131)* • Comparison function *(see page 135)*
9	Optionally, select an output to use as the **Reflex Output**. This reflex output is used when the Stop event *(see page 143)* is triggered. **NOTE:** This option is only available for TM3XF• expansion modules, which support external events.
10	Optionally, set the value of the **Stop → Stop Event** parameter to **Yes** to enable an external event *(see page 143)* when the counter stops at 0. In **Stop Event Name**, enter the name of the external event to reference in a task when the Stop event is triggered. **NOTE:** This option is only available for TM3XF• expansion modules, which support external events.

Programming the Main Type

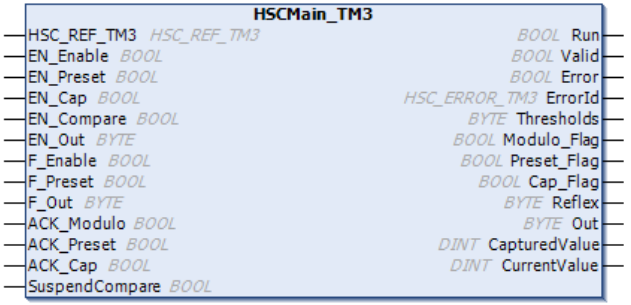
Overview

The **Main** type is always managed by an `HSCMain_TM3` *(see page 164)* function block.

NOTE: At build time, an error is detected if the `HSCMain_TM3` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3_HSC → HSC → HSCMain_TM3** in the list.
2	Drag-and-drop the item onto the **POU** window.
3	Edit the default **Main** type instance name to match the **Instance name** of the counter function block defined in the Configuration window.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **One-shot** mode.

This table describes the input variables:

Input	Type	Description
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC instance.
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes enabling of the counter with the Enable input *(see page 128)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter Preset via the Sync input *(see page 126)*.
EN_Cap	BOOL	When **CAP** input is configured: if TRUE, enables the Capture input.
EN_Compare	BOOL	TRUE = enables the comparator operation *(see page 135)* (using Thresholds 0, 1, 2, 3): • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1, Reflex2, Reflex3 output bits) • events (to trigger external tasks on threshold crossing) **NOTE:** This option is only available for TM3XF• expansion modules, which support external events.
EN_Out	BYTE	Set bits to 1 to enable the corresponding physical outputs to echo the configured function value (Reflex or Stop) as a result of the comparison function. Only active when outputs configured in HSC editor: • Bit 0: Out0 enabled. • Bit 1: Out1 enabled. • Bit 2: Out2 enabled. • Bit 3: Out3 enabled. • Bits 4...7: Not used. Association of HSC output Outx with physical output Qy is done by configuration.
F_Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
F_Preset	BOOL	On rising edge, initializes the counter. In Event counter and Frequency Meter modes, if enabled, restarts the internal timer relative to the time base.

Input	Type	Description
F_Out	BYTE	Set bits to 1 to force the corresponding physical outputs to 1 if associated with HSC by configuration. Takes priority over EN_Out. Only active when outputs configured in HSC editor: ● Bit 0: Output 0 forced. ● Bit 1: Output 1 forced. ● Bit 2: Output 2 forced. ● Bit 3: Output 3 forced. ● Bits 4...7: Not used. Association of HSC output Outx with physical output Qy is done by configuration.
ACK_Modulo	BOOL	No effect in one-shot mode.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets Cap_Flag.
SuspendCompare	BOOL	TRUE = compare results are suspended: ● Threshold, Reflex, and Out bits of the function block maintain their last value. ● Events are masked. **NOTE:** EN_Compare, EN_Reflex, and F_Out remain operational while SuspendCompare is set.

This table describes the output variables:

Output	Type	Comment
Run	BOOL	TRUE = counter is activated. Set to FALSE when the counter value reaches 0. A preset is needed to restart the counter
Valid	BOOL	TRUE = indicates that output values on the function block are valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See the HSC_ERROR_TM3 *(see page 152)* enumeration.
Thresholds	BYTE	Bits set to 1 when CurrentValue ≥ Threshold *(see page 135)* for corresponding threshold: • Bit 0: CurrentValue ≥ Threshold 0 • Bit 1: CurrentValue ≥ Threshold 1 • Bit 2: CurrentValue ≥ Threshold 2 • Bit 3: CurrentValue ≥ Threshold 3 • Bits 4...7: Not used Only active when EN_Compare is set.
Modulo_Flag	BOOL	Not used.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 126)*.
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 132)*. This flag must be reset before a new capture can occur.
Reflex	BYTE	State of the reflex outputs: • Bit 0: Reflex 0 • Bit 1: Reflex 1 • Bit 2: Reflex 2 • Bit 3: Reflex 3 • Bits 4...7: Not used
Out	BYTE	State of the physical outputs: • Bit 0: Output 0 • Bit 1: Output 1 • Bit 2: Output 2 • Bit 3: Output 3 • Bits 4...7: Not used Association of HSC output Outx with physical output Qy is done by configuration.
CapturedValue	DINT	Captured value, valid when Cap_Flag is TRUE.
CurrentValue	DINT	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the HSCGetParam_TM3 *(see page 171)* or HSCSetParam_TM3 *(see page 169)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M262 Logic/Motion Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the HSC_PARAMETER_TYPE_TM3 *(see page 155)* which can be read or modified while the program is running:

Parameter	Description
HSC_PRESET	To get or set the Preset value of an HSC
HSC_THRESHOLD0	To get or set the Threshold 0 value of an HSC
HSC_THRESHOLD1	To get or set the Threshold 1 value of an HSC
HSC_THRESHOLD2	To get or set the Threshold 2 value of an HSC
HSC_THRESHOLD3	To get or set the Threshold 3 value of an HSC
HSC_REFLEX0	To get or set output 0 reflex mode of an HSC function. The four least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 => TH0
HSC_REFLEX1	To get or set output 1 reflex mode of an HSC function. The four least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 => TH0
HSC_REFLEX2	To get or set output 2 reflex mode of an HSC function. The four least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 => TH0
HSC_REFLEX3	To get or set output 3 reflex mode of an HSC function. The four least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 => TH0

Part III
Modulo-loop Mode

Overview

This part describes the use of a HSC in **Modulo-loop** mode.

What Is in This Part?

This part contains the following chapters:

Chapter 6
Modulo-loop Principle

Modulo-loop Mode Principle Description

Overview

The **Modulo-loop** mode can be used for repeated actions on a series of moving objects, such as packaging and labeling applications.

Principle

On a rising edge of the Sync condition *(see page 126)*, the counter is activated and the counter value is set to 0.

When counting is enabled *(see page 128)*:

Incrementing direction: the counter increments until it reaches the modulo value -1. At the next pulse, the counter is reset to 0, a modulo flag is set to 1, and the counting continues.

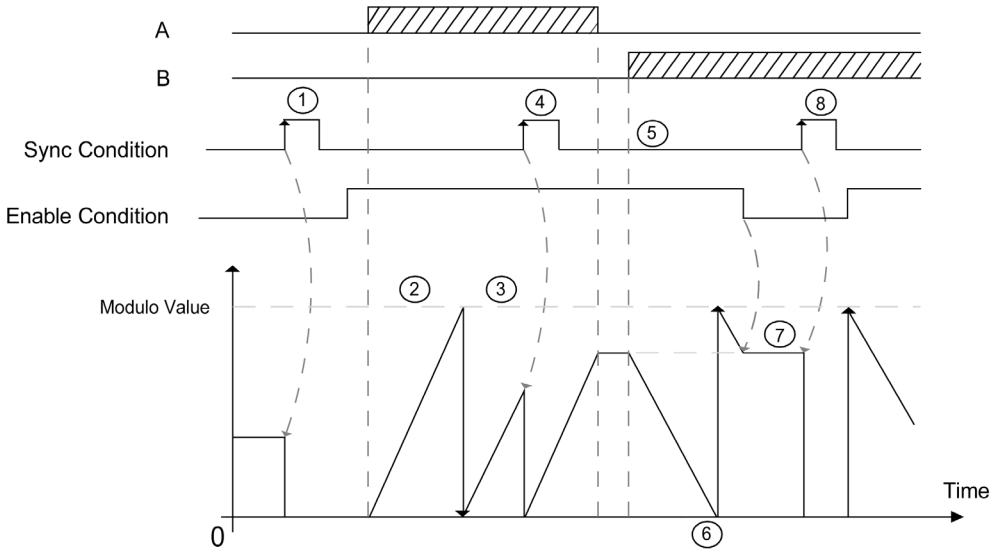
Decrementing direction: the counter decrements until it reaches 0. At the next pulse, the counter is set to the modulo value, a modulo flag is set to 1, and the counting continues.

Input Modes

This table shows the 8 types of input modes available:

Input Mode	Comment
A = Up, B = Down	default mode The counter increments on A and decrements on B.
A = Impulse, B = Direction	If there is a rising edge on A and B is true, then the counter decrements. If there is a rising edge on A and B is false, then the counter increments.
Normal Quadrature X1	A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction: ● X1: 1 count by Encoder cycle ● X2: 2 counts by Encoder cycle ● X4: 4 counts by Encoder cycle
Normal Quadrature X2	
Normal Quadrature X4	
Reverse Quadrature X1	
Reverse Quadrature X2	
Reverse Quadrature X4	

Up Down Principle Diagram

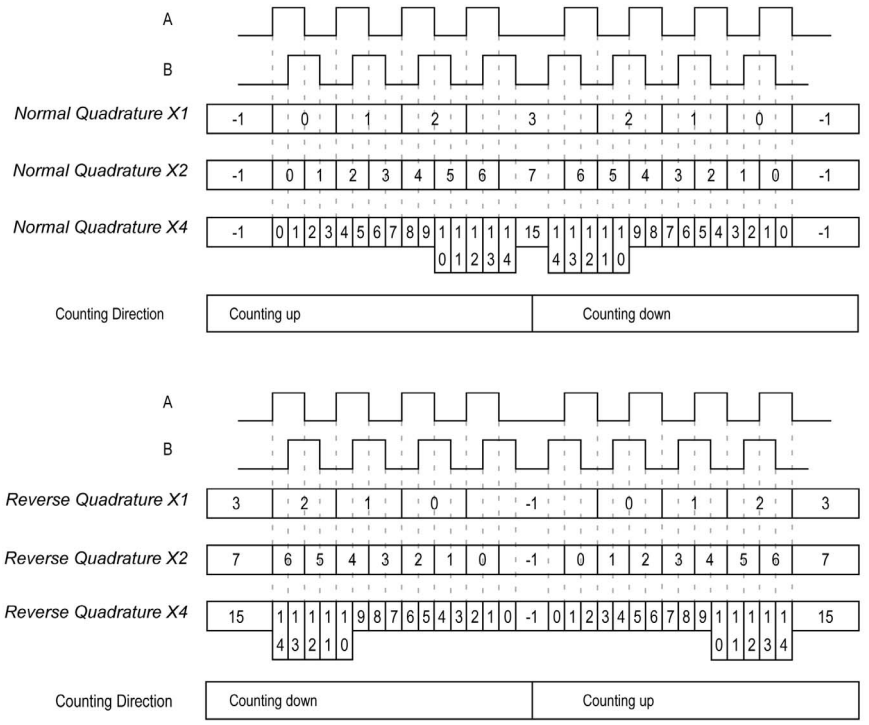


Stage	Action
1	On the rising edge of Sync condition, the counter value is reset to 0 and the counter is activated.
2	When Enable condition = 1, each pulses on A increments the counter value.
3	When the counter reaches the (modulo-1) value, the counter loops to 0 at the next pulse and the counting continues. `Modulo_Flag` is set to 1.
4	On the rising edge of Sync condition, the counter value is reset to 0.
5	When Enable condition = 1, each pulse on B decrements the counter.
6	When the counter reaches 0, the counter loops to (modulo-1) at the next pulse and the counting continues.
7	When Enable condition = 0, the pulses on the inputs are ignored.
8	On the rising edge of Sync condition, the counter value is reset to 0.

NOTE: Enable and Sync conditions depends on configuration. These are described in the Enable *(see page 128)* and Preset *(see page 126)* function.

Quadrature Principle Diagram

The encoder signal is counted according to the input mode selected, as shown below:



Chapter 7
Modulo-loop with a Simple Type

Overview

This chapter describes how to implement a High Speed Counter in **Modulo-loop** mode using a **Simple** type.

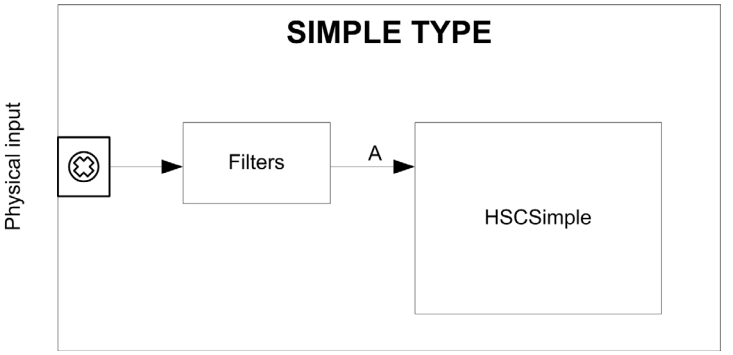
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Simple** type in **Modulo-loop** mode:



A **Simple** type counting for **Modulo-loop** mode only counts up.

Configuration of the Simple Type in Modulo-Loop Mode

Procedure

Follow this procedure to configure a **Simple** type in **Modulo-loop** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: The **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Simple**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Modulo-loop**.
5	In **Counting Inputs → A input → Location** select the input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
7	Enter the value of the **Range → Modulo** parameter to set the counting modulo value.

Programming the Simple Type

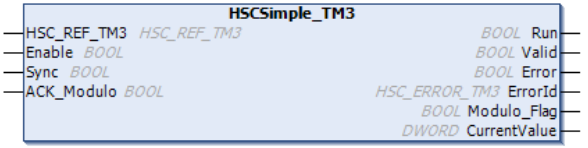
Overview

A **Simple** type is always managed by an `HSCSimple_TM3` *(see page 162)* function block.

NOTE: At build time, an error is detected if the `HSCSimple_TM3` function block is used to manage a different HSC type.

Adding a HSCSimple Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3_HSC → HSC → HSCSimple_TM3** in the list.
2	Drag-and-drop the item onto the **POU** window.
3	Edit the default **Simple** type instance name to match the **Instance name** of the counter function block defined in the Configuration window.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Modulo-loop** mode.

This table describes the input variables:

Input	Type	Comment
`HSC_REF_TM3`	`HSC_REF_TM3`	Reference to the HSC instance.
`Enable`	`BOOL`	`TRUE` = activates counter and takes into account pulses on the counter input.
`Sync`	`BOOL`	On rising edge, resets and initializes the counter.
`ACK_Modulo`	`BOOL`	On rising edge, resets `Modulo_Flag`.

This table describes the output variables:

Output	Type	Comment
Run	BOOL	TRUE = indicates counter is activated.
Valid	BOOL	TRUE = indicates that output values on the function block are valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See the HSC_ERROR_TM3 *(see page 152)* enumeration.
Modulo_Flag	BOOL	Set to TRUE when the counter rolls over the Modulo value.
CurrentValue	DWORD	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the `HSCGetParam_TM3` *(see page 171)* or `HSCSetParam_TM3` *(see page 169)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M262 Logic/Motion Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the `HSC_PARAMETER_TYPE_TM3` *(see page 155)* that can be read or modified while the program is running:

Parameter	Description
`HSC_MODULO`	To get or set the modulo value of an HSC.

Chapter 8
Modulo-loop with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Modulo-loop** mode using a **Main** type.

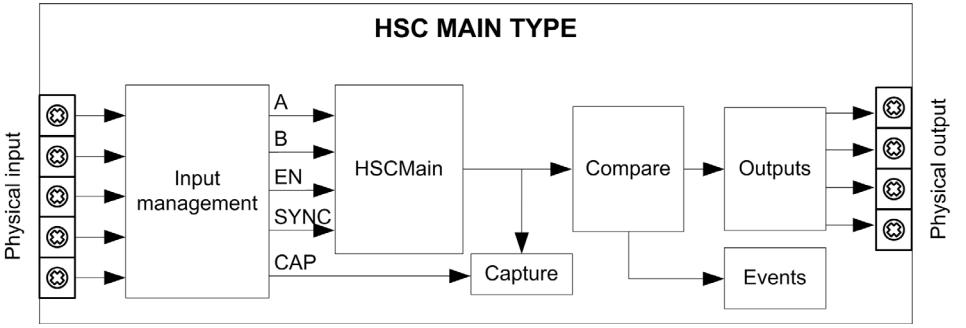
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **Modulo-loop** mode:



A and B are the counting inputs of the counter.

EN is the enable input of the counter.

SYNC is the synchronization input of the counter.

CAP is the capture input of the counter.

Optional Functions

In addition to the **Modulo-loop** mode, the **Main** type can provide the following functions:

- Enable function *(see page 128)*
- Capture function *(see page 131)*
- Comparison function *(see page 135)*

NOTE: The Preset value is 0 and cannot be modified.

Configuration of the Main Type Single Phase in Modulo-Loop Mode

Procedure

Follow this procedure to configure a **Main** type single phase in **Modulo-loop** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Single Phase**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Modulo-loop**.
5	In **Counting Inputs → A input → Location** select the input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
7	Enter the value of the **Range → Modulo** parameter to set the counting modulo value.
8	Optionally, you can enable these control functions: ● Enable function *(see page 128)* ● Capture function *(see page 131)* ● Preset function *(see page 126)* ● Comparison function *(see page 135)*

Configuration of the Main Type Dual Phase in Modulo-Loop Mode

Procedure

Follow this procedure to configure a **Main** type dual phase in **Modulo-loop** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Dual Phase**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Modulo-loop**.
5	Set the value of the **General → Input mode** parameter to select the modulo loop input mode *(see page 51)*.
6	In **Counting Inputs → A input → Location** select the input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
7	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
8	In **Counting Inputs → B input → Location** select the input to use as the B input.
9	Set the value of the **Counting inputs → B input → Filter** parameter to reduce the bounce effect on the input..
10	Enter the value of the **Range → Modulo** parameter to set the counting modulo value.
11	Optionally, you can enable these control functions: ● Preset function *(see page 126)* ● Capture function *(see page 131)* ● Comparison function *(see page 135)*

Programming the Main Type

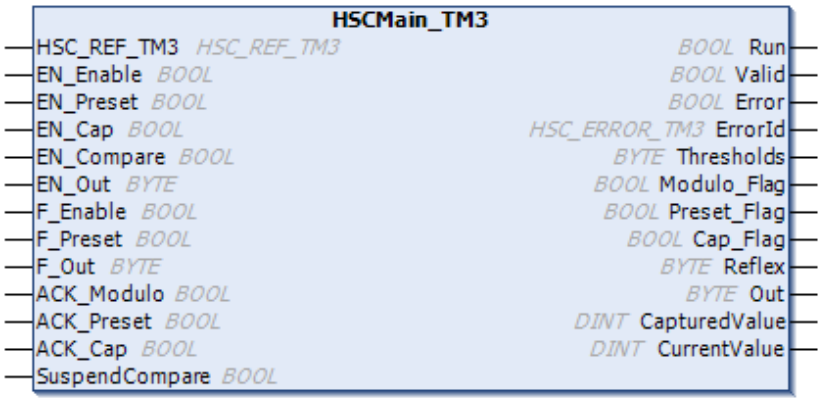
Overview

The **Main** type is always managed by an `HSCMain_TM3` *(see page 164)* function block.

NOTE: At build time, an error is detected if the `HSCMain_TM3` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3_HSC → HSC → HSCMain_TM3** in the list.
2	Drag-and-drop the item onto the **POU** window.
3	Edit the default **Main** type instance name to match the **Instance name** of the counter function block defined in the Configuration window.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Modulo-loop** mode.

This table describes the input variables:

Input	Type	Description
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC instance.
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes the counter enable via the Enable input *(see page 128)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter to be initialized via the Preset input.
EN_Cap	BOOL	When **CAP** input is configured: if TRUE, enables the Capture input.
EN_Compare	BOOL	TRUE = enables the comparison function *(see page 135)* using Threshold 0, 1, 2, 3: • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1, Reflex2, Reflex3 output bits) • events (to trigger external tasks on threshold crossing)
EN_Out	BYTE	Set bits to 1 of corresponding physical outputs to echo the configured function value (Reflex or Stop) as a result of the comparison function. Only active when outputs configured in HSC editor: • Bit 0: Output 0 enabled. • Bit 1: Output 1 enabled. • Bit 2: Output 2 enabled. • Bit 3: Output 3 enabled. • Bits 4...7: Not used. Association of HSC output Outx with physical output Qy is done by configuration.
F_Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
F_Preset	BOOL	On rising edge, resets, and initializes the counter.
F_Out	BYTE	Set bits to 1 to force corresponding physical outputs to 1 if associated with HSC by configuration. Takes priority over EN_Out. Only active when outputs configured in HSC editor: • Bit 0: Output 0 forced. • Bit 1: Output 1 forced. • Bit 2: Output 2 forced. • Bit 3: Output 3 forced. • Bits 4...7: Not used. Association of HSC output Outx with physical output Qy is done by configuration.
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag.

Input	Type	Description
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets Cap_Flag.
SuspendCompare	BOOL	TRUE = compare results are suspended: ● Threshold, Reflex and Out bits maintain their last value. ● Events are masked. **NOTE:** EN_Compare, EN_Reflex, and F_Out remain operational while SuspendCompare is set.

This table describes the output variables:

Output	Type	Comment
Run	BOOL	TRUE = counter is activated. Only set to FALSE if the counter is disabled or an error is detected.
Valid	BOOL	TRUE = indicates that output values on the function block are valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See HSC_ERROR_TM3 *(see page 152)* enumeration.
Thresholds	BYTE	Bits set to 1 when CurrentValue ≥Threshold *(see page 135)*: ● Bit 0: CurrentValue ≥ Threshold 0 ● Bit 1: CurrentValue ≥ Threshold 1 ● Bit 2: CurrentValue ≥ Threshold 2 ● Bit 3: CurrentValue ≥ Threshold 3 ● Bits 4...7: Not used Only active when EN_Compare is set.
Modulo_Flag	BOOL	Set to 1 when the counter roll overs the modulo or 0.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 126)*.
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 132)*. This flag must be reset before a new capture can occur.
Reflex	BYTE	State of the reflex function: ● Bit 0: Reflex 0 ● Bit 1: Reflex 1 ● Bit 2: Reflex 2 ● Bit 3: Reflex 3 ● Bits 4...7: Not used

Output	Type	Comment
Out	BYTE	State of the physical outputs: ● Bit 0: Output 0 ● Bit 1: Output 1 ● Bit 2: Output 2 ● Bit 3: Output 3 ● Bits 4...7: Not used Association of output Outx with physical output Qy is done by configuration.
CapturedValue	DINT	Captured value, valid when Cap_Flag is TRUE.
CurrentValue	DINT	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the HSCGetParam_TM3 *(see page 171)* or HSCSetParam_TM3 *(see page 169)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M262 Logic/Motion Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the HSC_PARAMETER_TYPE_TM3 *(see page 155)* that can be read or modified while the program is running:

Parameter	Description
HSC_MODULO	To get or set the Modulo value of an HSC.
HSC_THRESHOLD0	To get or set the Threshold 0 value of an HSC.
HSC_THRESHOLD1	To get or set the Threshold 1 value of an HSC.
HSC_THRESHOLD2	to get or set the Threshold 2 value of an HSC
HSC_THRESHOLD3	to get or set the Threshold 3 value of an HSC
HSC_REFLEX0	To get or set the output 0 reflex mode. The five least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 =< TH0.
HSC_REFLEX1	To get or set the output 1 reflex mode. The five least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 =< TH0.
HSC_REFLEX2	To get or set the output 2 reflex mode. The five least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 =< TH0.
HSC_REFLEX3	To get or set the output 3 reflex mode. The five least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 =< TH0.

Part IV
Free-large Mode

Overview

This part describes the use of an HSC in **Free-large** mode.

What Is in This Part?

This part contains the following chapters:

Chapter 9
Free-large Mode Principle

Overview

This chapter describes the principle of the **Free-large** mode.

What Is in This Chapter?

This chapter contains the following topics:

Free-large Mode Principle Description

Overview

The **Free-large** mode can be used for axis monitoring or labeling in cases where the incoming position of each part has to be known.

Principle

In the **Free-large** mode, the module behaves like a standard up and down counter.

When counting is enabled *(see page 128)*, the counter counts as follows in:
Incrementing direction: the counter increments.
Decrementing direction: the counter decrements.

The counter is activated by a preset edge which loads the preset value.

The current counter is stored in the capture register by using the Capture *(see page 131)* function.

If the counter reaches the counting limits, the counter will react according to the Limits Management *(see page 77)* configuration.

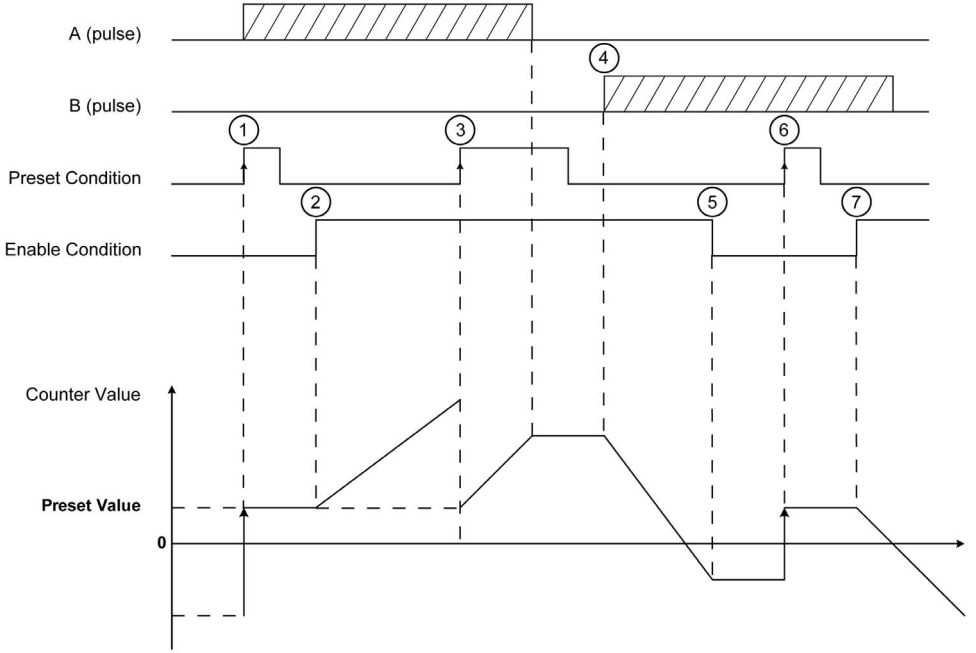
Input Modes

This table shows the 8 types of input modes available:

Input Mode	Comment
A = Up, B = Down	default mode The counter increments on A and decrements on B.
A = Pulse, B = Direction	If there is a rising edge on A and B is true, then the counter decrements. If there is a rising edge on A and B is false, then the counter increments.
Normal Quadrature X1	A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction: ● X1: 1 count for each Encoder cycle ● X2: 2 counts for each Encoder cycle ● X4: 4 counts for each Encoder cycle
Normal Quadrature X2	
Normal Quadrature X4	
Reverse Quadrature X1	
Reverse Quadrature X2	
Reverse Quadrature X4	

Up Down Principle Diagram

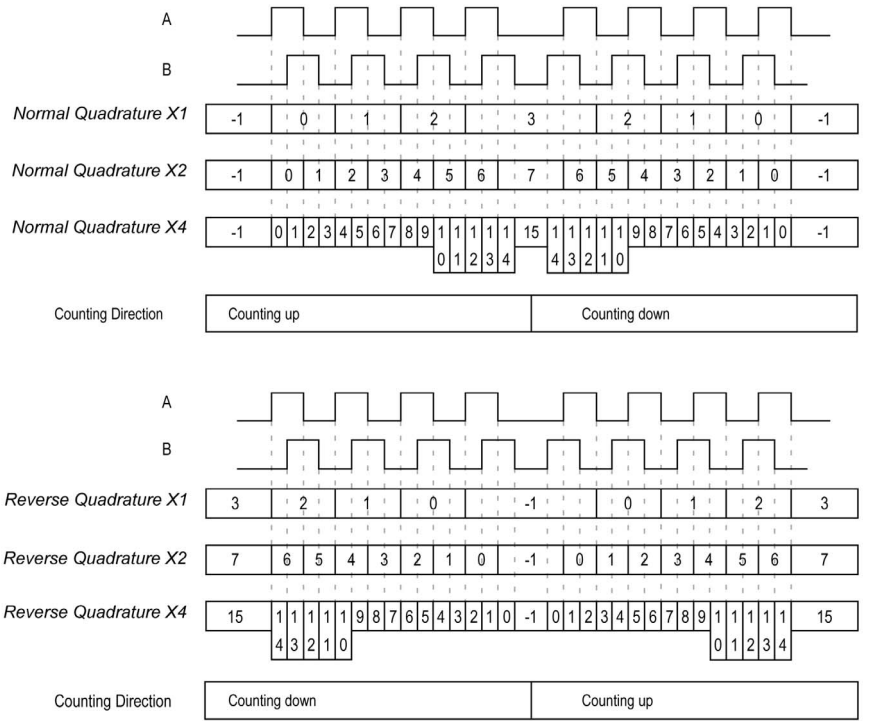
The figures shows the **A = Up, B = Down** mode:



Stage	Action
1	On the rising edge of Preset condition, the counter value is set to the preset value and the counter is activated.
2	When Enable condition = 1, each pulse on A increment the counter value.
3	On the rising edge of Preset condition, the counter value is set to the preset value.
4	When Enable condition = 1, each pulse on B decrements the counter value.
5	When Enable condition = 0, the pulses on A or B are ignored.
6	On the rising edge of Preset condition, the counter value is set to the preset value.
7	When Enable condition = 1, the pulses on B decrements the counter value.

Quadrature Principle Diagram

The encoder signal is counted according to the input mode selected, as shown below:



Limits Management

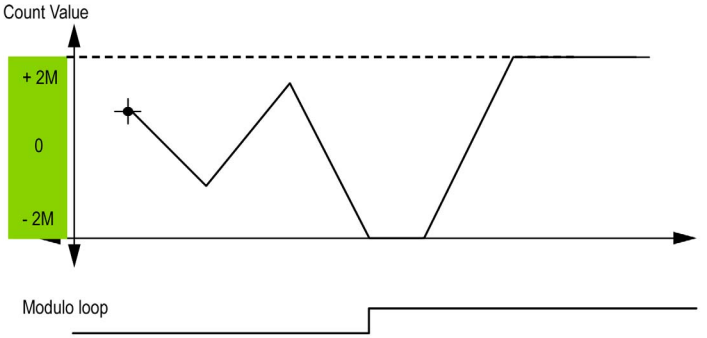
Overview

In **Free-large** mode, when the counter limit is reached, the counter can have 2 behaviors depending on configuration:

- Lock on limits
- Rollover

Lock on Limits

In the case of an overflow or underflow counter, the counter value is maintained at the limit value, and the modulo loop value goes to 1.



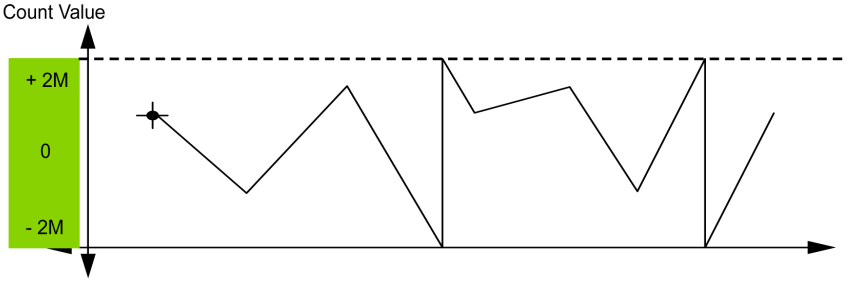
2M value is given as:

- +2M = 2 (exp 31) -1
- -2M = -2 (exp 31)

Rollover

In the case of overflow or underflow of the counter, the counter value goes automatically to the opposite limit value.

`Modulo_Flag` output is set to 1.



Chapter 10
Free-large with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Free-large** mode using a **Main** type.

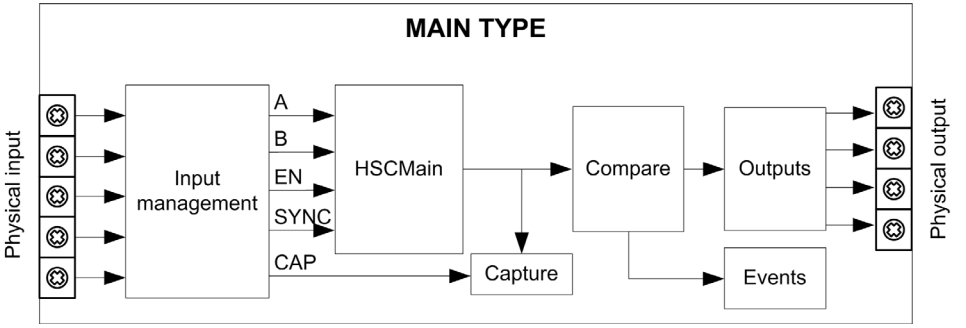
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **Free-large** mode:



A and B are the counting inputs of the counter.

EN is the enable input of the counter.

SYNC is the synchronization input of the counter.

CAP is the capture input of the counter.

Optional Function

In addition to the **Free-large** mode, the **Main** type can provide the following functions:

- Preset function *(see page 126)*
- Enable function *(see page 128)*
- Capture function *(see page 131)*
- Comparison function *(see page 135)*

Configuration of the Main Type Dual Phase in Free-Large Mode

Procedure

Follow this procedure to configure a **Main** type dual phase in **Free-large** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Dual Phase**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Free-large**.
5	Set the value of the **General → Input mode** parameter to select the input mode *(see page 74)*.
6	In **Counting Inputs → A input → Location** select the input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
7	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
8	In **Counting Inputs → B input → Location** select the input to use as the B input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
9	Set the value of the **Counting inputs → B input → Filter** parameter.
10	Enter the value of the **Range → Preset** parameter to set the counting initial value.
11	Enter the value of the **Range → Limits** for limits management *(see page 77)*.
12	Optionally, you can enable these functions: ● Preset function *(see page 126)* ● Enable function *(see page 128)* ● Capture function *(see page 131)* ● Comparison function *(see page 135)*

Programming the Main Type

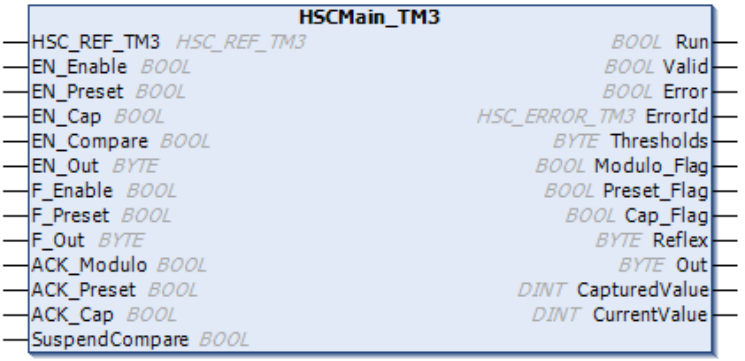
Overview

The **Main** type is always managed by an `HSCMain_TM3` *(see page 164)* function block.

NOTE: At build time, an error is detected if the `HSCMain_TM3` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3_HSC → HSC → HSCMain_TM3** in the list.
2	Drag-and-drop the item onto the **POU** window.
3	Edit the default **Main** type instance name to match the **Instance name** of the counter function block defined in the Configuration window.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Free-large** mode.

This table describes the input variables:

Input	Type	Description
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC instance.
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes the counter enable via the Enable input *(see page 128)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter Preset via the Sync input *(see page 126)*.
EN_Cap	BOOL	When **CAP** input is configured: if TRUE, enables the Capture input *(see page 133)*.
EN_Compare	BOOL	TRUE = enables the comparison operation *(see page 135)* (using Thresholds 0, 1, 2, 3): • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1 output bits) • events (to trigger external tasks on threshold crossing) **NOTE:** This option is only available for TM3XF• expansion modules, which support external events.
EN_Out	BYTE	Set bits to 1 to enable corresponding physical outputs to echo the configured function value (Reflex or Stop) as a result of the comparison function. Only active when outputs configured in HSC editor: • Bit 0: Output 0 enabled. • Bit 1: Output 1 enabled. • Bit 2: Output 2 enabled. • Bit 3: Output 3 enabled. • Bits 4...7: Not used. Association of HSC output Outx with physical output Qy is done by configuration.
F_Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
F_Preset	BOOL	On rising edge, presets and initializes the counter.

Input	Type	Description
F_Out	BYTE	Set bits to 1 to force corresponding physical outputs to 1 if associated with HSC by configuration. Takes priority over EN_Out. Only active when outputs configured in HSC editor: ● Bit 0: Output 0 forced. ● Bit 1: Output 1 forced. ● Bit 2: Output 2 forced. ● Bit 3: Output 3 forced. ● Bits 4...7: Not used. Association of HSC output Outx with physical output Qy is done by configuration.
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets Cap_Flag.
SuspendCompare	BOOL	TRUE = compare results are suspended: ● Threshold and Reflex output bits maintain their last value ● Physical outputs maintain their last value. ● Events are masked. **NOTE:** EN_Compare, EN_Reflex, F_Out remain operational while SuspendCompare is set.

This table describes the output variables:

Outputs	Type	Comment
Run	BOOL	TRUE = counter is activated. Set to 1 at first counter preset. Set to 0 if the counter is disabled or an error is detected.
Valid	BOOL	TRUE = indicates that output values on the function block are valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See HSC_ERROR_TM3 *(see page 152)* enumeration.
Thresholds	BYTE	Bits set to 1 when CurrentValue ≥ Threshold *(see page 135)*: ● Bit 0: CurrentValue ≥ Threshold 0 ● Bit 1: CurrentValue ≥ Threshold 1 ● Bit 2: CurrentValue ≥ Threshold 2 ● Bit 3: CurrentValue ≥ Threshold 3 ● Bits 4...7: Not used Only active when EN_Compare is set.

Outputs	Type	Comment
Modulo_Flag	BOOL	Set to 1 when the counter rolls over its limits.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 126)*
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 131)*. This flag must be reset before a new capture can occur.
Reflex	BYTE	State of the reflex function: ● Bit 0: Reflex 0 ● Bit 1: Reflex 1 ● Bit 2: Reflex 2 ● Bit 3: Reflex 3 ● Bits 4...7: Not used
Out	BYTE	State of the physical outputs: ● Bit 0: Output 0 ● Bit 1: Output 1 ● Bit 2: Output 2 ● Bit 3: Output 3 ● Bits 4...7: Not used Association of HSC output Outx with physical output Qy is done by configuration.
CapturedValue	DINT	Captured value, valid when Cap_Flag is TRUE.
CurrentValue	DINT	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the HSCGetParam_TM3 *(see page 171)* or HSCSetParam_TM3 *(see page 169)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M262 Logic/Motion Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the HSC_PARAMETER_TYPE_TM3 *(see page 155)* enumeration which can be read or modified while the program is running:

Parameter	Description
HSC_PRESET	To get or set the Preset value of an HSC.
HSC_THRESHOLD0	To get or set the Threshold 0 value of an HSC.
HSC_THRESHOLD1	To get or set the Threshold 1 value of an HSC.
HSC_THRESHOLD2	to get or set the Threshold 2 value of an HSC
HSC_THRESHOLD3	to get or set the Threshold 3 value of an HSC
HSC_REFLEX0	To get or set the output 0 reflex mode. The four least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 =< TH0.
HSC_REFLEX1	To get or set the output 1 reflex mode. The four least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 =< TH0.
HSC_REFLEX2	To get or set the output 2 reflex mode. The four least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 =< TH0.
HSC_REFLEX3	To get or set the output 3 reflex mode. The four least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 =< TH0.

Part V
Event Counting Mode

Overview

This part describes the use of an HSC in **Event Counting** mode.

What Is in This Part?

This part contains the following chapters:

Chapter 11
Event Counting Principle

Event Counting Mode Principle Description

Overview

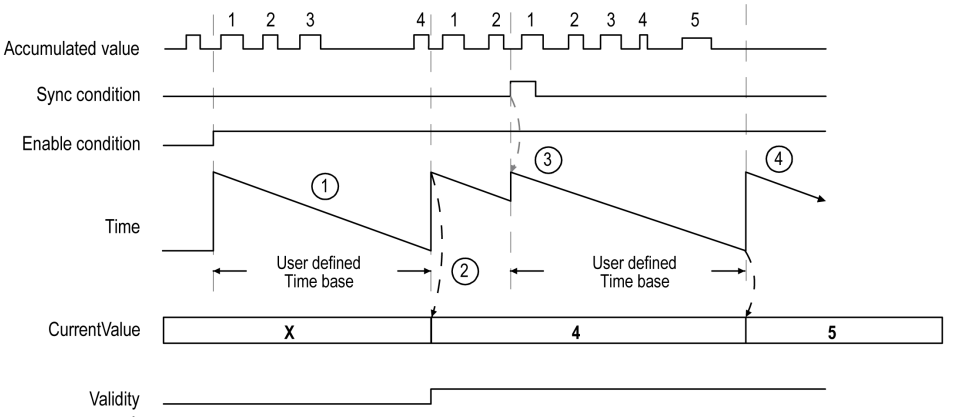
The **Event Counting** mode allows you to count the number of events that occur during a given period of time.

Principle

The counter assesses the number of pulses applied to the input for a predefined period of time. At the end of each period, the counting register is updated with the number of events received.

Synchronization can be used over the time period. This restarts the counting event for a new predefined time period. The counting restarts at the edge Sync condition *(see page 126)*.

Principle Diagram



Stage	Action
1	When Enable condition = 1, the counter accumulates the number of events (pulses) on the physical input during a predefined period of time. If Validity = 0, the counter value is not relevant.
2	Once the first period of time has elapsed, the counter value is set to the number of events counted over the period and Validity is set to 1. The counting restarts for a new period of time.
3	On the rising edge of the Sync condition: ● the accumulated value is reset to 0 ● the counter value is not updated ● the counting restarts for a new period of time
4	Once the period of time has elapsed, the counter value is set to the number of events counted over the period. The counting restarts for a new period of time.

NOTE:
On the **Main** type, when the Enable condition is:

- Set to 0: the current counting is aborted and `CurrentValue` is maintained at the previous valid value.
- Set to 1: the accumulated value is reset to 0, the `CurrentValue` remains unchanged, and the counting restarts for a new period of time.

Chapter 12
Event Counting with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Event Counting** mode using a **Main** type.

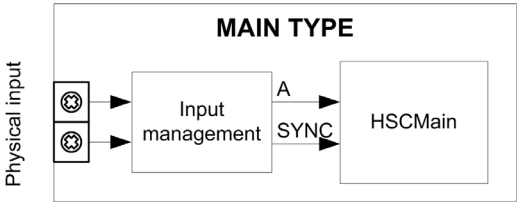
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **Event Counting** mode.



A is the counting input of the counter.

SYNC is the synchronization input of the counter.

Optional Function

In addition to the **Event Counting** mode, the **Main** type provides the Preset function *(see page 126)*.

Configuration of the Main Type Single Phase in Event Counting Mode

Procedure

Follow this procedure to configure a **Main** type single phase in **Event Counting** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **HSC Main Single Phase**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → Counting Mode** parameter to **Event Counting**.
5	In **Counting Inputs → A input → Location** select the input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
7	Set the value of the **Range → Time base** parameter to determine the period during which the number of events is counted. Select the measurement of the update cycle time: ● 0.1 s ● 1 s (default value) ● 10 s ● 60 s
8	Optionally, set the value of the **Control inputs → SYNC input → Location** parameter to enable the Preset Function *(see page 126)*.

Programming the Main Type

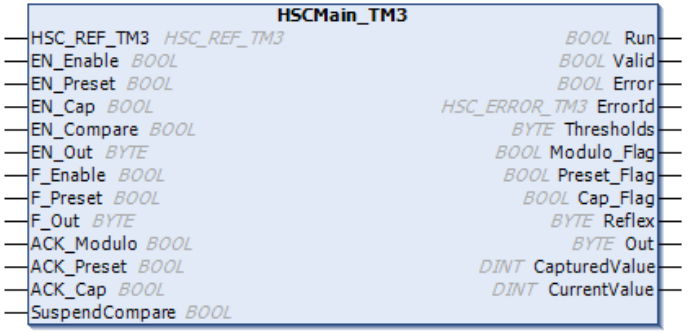
Overview

The **Main** type is always managed by an `HSCMain_TM3` *(see page 164)* function block.

NOTE: At build time, an error is detected if the `HSCMain_TM3` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3_HSC → HSC → HSCMain_TM3** in the list.
2	Drag-and-drop the item onto the **POU** window.
3	Edit the default **Main** type instance name to match the **Instance name** of the counter function block defined in the Configuration window.



I/O Variables Usage

These tables describe how the different pins of the function block are used in the mode **Event**.

This table describes the input variables:

Input	Type	Description
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC instance.
EN_Enable	BOOL	Not used.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter Preset via the Sync input *(see page 126)*.
EN_Cap	BOOL	Not used.
EN_Compare	BOOL	Not used.
EN_Out	BYTE	Not used.
F_Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
F_Preset	BOOL	On rising edge, restarts the internal timer relative to the time base.
F_Out	BYTE	Not used.
ACK_Modulo	BOOL	Not used.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	Not used.
SuspendCompare	BOOL	Not used.

This table describes the output variables:

Outputs	Type	Comment
Run	BOOL	TRUE = counter is activated.
Valid	BOOL	TRUE = CurrentValue is valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See the HSC_ERROR_TM3 *(see page 152)* enumeration.
Thresholds	BYTE	Not used.
Modulo_Flag	BOOL	Not used.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 126)*.
Cap_Flag	BOOL	Not used.
Reflex	BYTE	Not used.
Out	BYTE	Not used.
CapturedValue	DINT	Not used.
CurrentValue	DINT	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the HSCGetParam_TM3 *(see page 171)* or HSCSetParam_TM3 *(see page 169)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M262 Logic/Motion Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the HSC_PARAMETER_TYPE_TM3 *(see page 155)* which can be read or modified while the program is running:

Parameter	Type	Description
HSC_TIMEBASE	HSC_EVENT_TIMEBASE_TYPE_TM3 *(see page 153)*	To get or set the Timebase value of the HSC.

Part VI
Frequency Meter Type

Overview

This part describes the use of an HSC in **Frequency meter** type.

What Is in This Part?

This part contains the following chapters:

Chapter 13
Frequency Meter Principle

Description

Overview

The **Frequency meter** type measures an event frequency in Hz.

The **Frequency meter** type calculates the number of pulses in time intervals of 1 s. An updated value in Hz is available for each time base value (10, 100, or 1000 ms).

When there is a variation in the frequency, the value restoration time is 1 s with a value precision of 1 Hz.

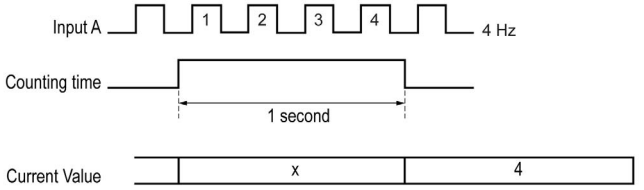
Operation Limits

The maximum frequency that the module can measure on the A input is 200 kHz. Beyond 200 kHz, the counting register value may decrease until it reaches 0.

The maximum duty cycle at 200 kHz is 55%.

Synopsis Diagram

This diagram provides an overview of the **Frequency meter** principle:



Chapter 14
Frequency Meter with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Frequency meter** mode with a **Main** type.

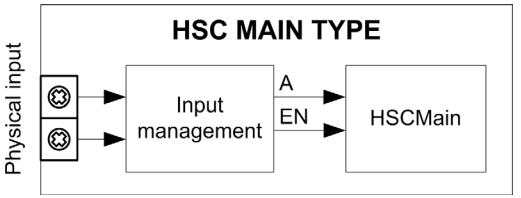
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main** type in **Frequency meter** type:



A is the counting input of the counter.

EN is the enable input of the counter.

Optional Function

In addition to the **Frequency meter** type, the **Main** type can provide the Enable function *(see page 128)*.

Configuration of the Frequency Meter Type

Procedure

Follow this procedure to configure a **Frequency Meter** type:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **Frequency Meter**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE:** **Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	In **Counting Inputs → A input → Location** select the input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
5	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
6	Set the value of the **Range → Time base** parameter to determine the period during which the number of events is counted. Select the measurement of the update cycle time: ● 10 ms ● 100 ms ● 1000 ms (default value)
7	Optionally, set the value of the **Control inputs → EN input → Location** parameter to enable the Enable Function *(see page 128)*.

Programming

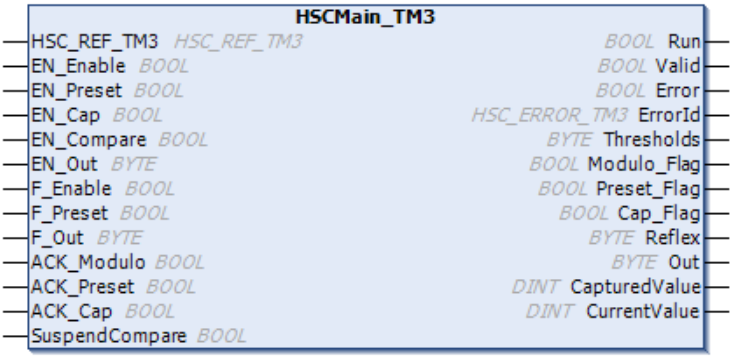
Overview

The **Main** type is always managed by an `HSCMain_TM3` *(see page 164)* function block.

NOTE: At build time, an error is detected if the `HSCMain_TM3` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3_HSC → HSC → HSCMain_TM3** in the list.
2	Drag-and-drop the item onto the **POU** window.
3	Edit the default **Main** type instance name to match the **Instance name** of the counter function block defined in the Configuration window.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Frequency meter** type.

This table describes the input variables:

Input	Type	Description
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC instance.
EN_Enable	BOOL	If TRUE and the **EN** input is configured, authorizes the counter to be enabled using the Enable input *(see page 128)*.
EN_Preset	BOOL	Not used.
EN_Cap	BOOL	Not used.
EN_Compare	BOOL	Not used.
EN_Out	BYTE	Not used.
F_Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
F_Preset	BOOL	On rising edge, restarts the internal timer relative to the time base. The CurrentValue is not preset.
F_Out	BYTE	Not used.
ACK_Modulo	BOOL	Not used.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	Not used.
SuspendCompare	BOOL	Not used

This table describes the output variables:

Outputs	Type	Comment
Run	BOOL	TRUE = counter is activated.
Valid	BOOL	Set to TRUE when CurrentValue is valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See the HSC_ERROR_TM3 *(see page 152)* enumeration.
Thresholds	BYTE	Not used.
Modulo_Flag	BOOL	Not used.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 126)*
Cap_Flag	BOOL	Not used.
Reflex	BYTE	Not used.
Out	BYTE	Not used.
CapturedValue	DINT	Not used.
CurrentValue	DINT	The value of the counter.

Part VII
Period Meter Type

Overview

This part describes the use of an HSC in **Period meter** type.

What Is in This Part?

This part contains the following chapters:

Chapter 15
Period Meter Type Principle

Description

Overview

Use the **Period meter** type to:
- Determine the duration of an event
- Determine the time between two events
- Set and measure the execution time for a process.

The **Period meter** can be used in two ways:
- Edge to opposite: Allows measurement of the duration of an event.
- Edge to edge: Allows measurement of the time between two events.

The measurement is expressed in the units defined by the **Resolution** parameter (0.1 µs, 1 µs, 100 µs, 1000 µs).

For example, if the counter value = 100 and the **Resolution** parameter is:
0.0001 (0.1 µs) measurement = 0.01 ms
0.001 (1 µs) measurement = 0.1 ms
0.1 (100 µs) measurement = 10 ms
1 (1000 µs) measurement = 100 ms

A timeout value can be specified in the configuration screen. Measurement is stopped if this timeout value is exceeded. In this case, the counting register is not valid until the next complete measurement.

The timeout value is defined in the same user-defined units (**Resolution**) as the measurement. When the timeout is set to 0, the timeout value is set to the maximum value.

Edge to Opposite Mode

The Edge to Opposite mode measures the duration of an event.

When the Enable condition = 1, the measurement is taken between the rising edge and the falling edge of the A input. The counting register is updated as soon as the falling edge is detected.

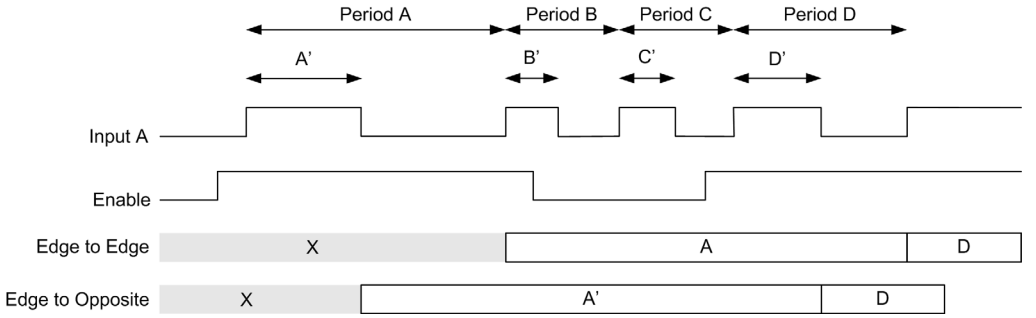
Edge to Edge Mode

The Edge to Edge mode measures the elapsed time between two events.

When the Enable condition = 1, the measurement is taken between two rising edges of the A input. The counting register is updated as soon as the second rising edge is detected.

Enable Condition Interruption Behavior

The trend diagram below describes the behavior of the counting register when the Enable condition is interrupted:



Threshold, Event, and Reflex Output Behavior

The values of up to 4 defined thresholds can be compared to the counter value and additional actions taken.

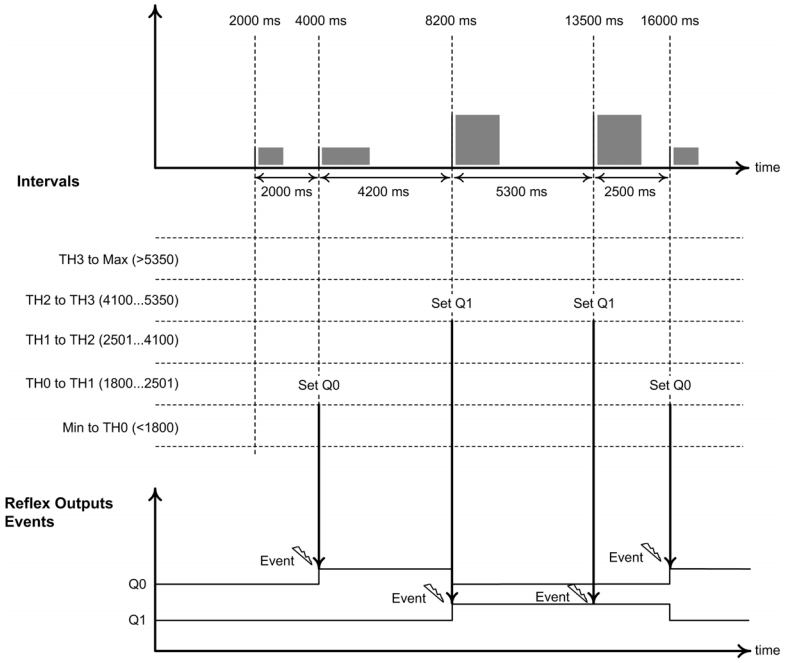
Reflex outputs can be set or reset when the counter value is:

- Below the smallest threshold value
- Between two threshold values
- Above the largest threshold value.

(TM3XF• expansion modules only) External events can be triggered when the counter value is:

- Below any threshold value
- Above any threshold value.
- Between two adjacent threshold values

This figure shows an example in Edge to Edge mode with 4 thresholds configured. Events are triggered and 2 reflex outputs activated (Q0 and Q1) when the measured value is between two threshold values:



Operating Limits

This table shows the equivalent measurement duration for each configurable **Resolution**:

Resolution	Max. Duration
0.1 µs	85,899 s
1 µs	429,496 s
100 µs	4,294,967 s
1000 µs	4,294,967 s

This table shows the maximum timeout value for each configurable **Resolution**:

Resolution	Max. Timeout Value
0.1 µs	858,993,459
1 µs	429,496,729
100 µs	42,949,672
1000 µs	4,294,967

Chapter 16
Period Meter with a Main Type

Overview

This chapter describes how to implement a High Speed Counter in **Period meter** mode with a **Main** type.

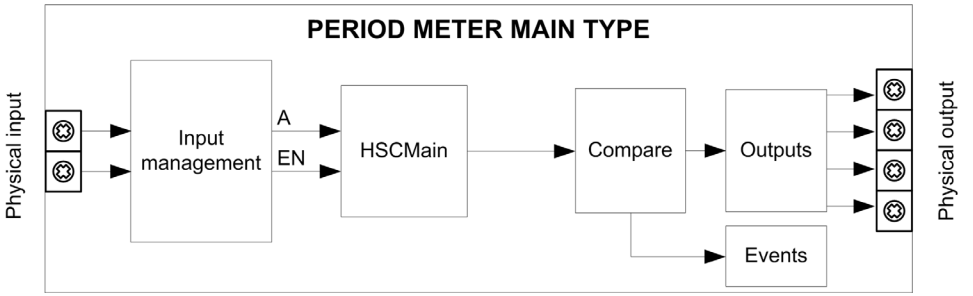
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

This diagram provides an overview of the **Main type** in **Period meter** type:



A is the counting input of the counter.

EN is the enable input of the counter.

Optional Function

In addition to the **Period meter** type, the **Main** type can provide the following function:

- Enable function *(see page 128)*

Configuration of the Period Meter Type in Edge to Edge Mode

Procedure

Follow this procedure to configure a **Period Meter** type in **Edge to Edge** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **Period Meter**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE:** **Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → PeriodMeter Mode** parameter to **Edge to Edge**.
5	In **Counting Inputs → A input → Location**, select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the inputs. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
7	Set the value of the **Range → Resolution** parameter. Select the unit of measurement: • 0.1 µs • 1 µs (default value) • 100 µs • 1000 µs
8	Enter the value of the **Range → Timeout** parameter to set the time value that a measured period must not exceed.
9	Optionally, you can enable these functions: • Enable function *(see page 128)* • Comparison function *(see page 135)*
10	Optionally, set the value of the **New Period Event** parameter to **Yes** to specify that an event is generated at the end of the configured time period. **NOTE:** This option is only available for TM3XF• expansion modules, which support external events. In **New Period Event Name**, type the name of the external event to use to reference the new period event in a task.

Configuration of the Period Meter Type in Edge to Opposite Mode

Procedure

Follow this procedure to configure a **Period Meter** type in **Edge to Opposite** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**. **Result**: **Counters** editor tab opens for HSC configuration. **NOTE:** A message appears at the bottom of the configuration screen if the maximum number of **HSC Main** functions has already been configured. Consider using an **HSC Simple** function instead.
2	In the **Counters** editor tab, set the value of the **Counting function** parameter to **Period Meter**. **Result**: The configuration parameters appear in the **Counters** editor tab.
3	If necessary, enter the value of the **General → Instance name** parameter. **NOTE: Instance name** is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the **General → PeriodMeter Mode** parameter to **Edge to Opposite**.
5	In **Counting Inputs → A input → Location**, select the regular or fast input to use as the A input. **NOTE:** A message is displayed at the bottom of the configuration window if no more I/Os are available for configuration. Free up one or more I/Os before continuing configuration of this function.
6	Set the value of the **Counting inputs → A input → Filter** parameter to reduce the bounce effect on the inputs. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
7	Set the value of the **Range → Resolution** parameter. Select the unit of measurement: ● 0.1 µs ● 1 µs (default value) ● 100 µs ● 1000 µs
8	Enter the value of the **Range → Timeout** parameter to set the time value that a measured period must not exceed.
9	Optionally, you can enable these functions: ● Enable function *(see page 128)* ● Comparison function *(see page 135)*
10	Optionally, set the value of the **New Period Event** parameter to **Yes** to specify that an event is generated at the end of the configured time period. **NOTE:** This option is only available for TM3XF• expansion modules, which support external events. In **New Period Event Name**, type the name of the external event to use to reference the new period event in a task.

Programming

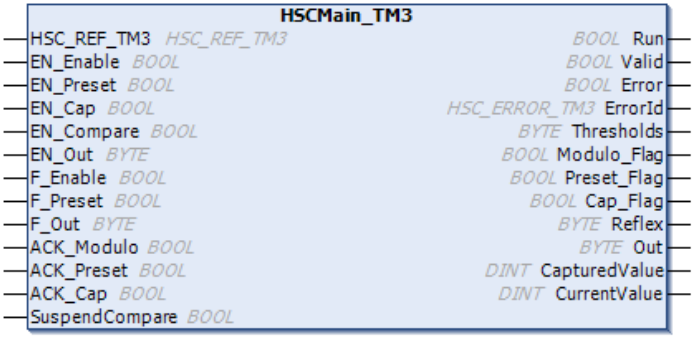
Overview

The **Main** type is always managed by an `HSCMain_TM3` *(see page 164)* function block.

NOTE: At build time, an error is detected if the `HSCMain_TM3` function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3_HSC → HSC → HSCMain_TM3** in the list.
2	Drag-and-drop the item onto the **POU** window.
3	Edit the default **Main** type instance name to match the **Instance name** of the counter function block defined in the Configuration window.



I/O Variables Usage

The tables below describe how the different pins of the function block are used in **Period meter** type.

This table describes the input variables:

Input	Type	Description
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC instance.
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes the counter enable via the Enable input *(see page 128)*.
EN_Preset	BOOL	Not used.
EN_Cap	BOOL	Not used.
EN_Compare	BOOL	TRUE = enables the comparison function *(see page 135)* (using Thresholds 0, 1, 2, 3): • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1 output bits) • events (to trigger external tasks on threshold crossing) **NOTE:** This option is only available for TM3XF• expansion modules, which support external events.
EN_Out	BYTE	Set bits to 1 to enable corresponding physical outputs to echo the configured function value (Reflex or Stop) as a result of the comparison function. Only active when outputs configured in HSC editor: • Bit 0: Output 0 enabled. • Bit 1: Output 1 enabled. • Bit 2: Output 2 enabled. • Bit 3: Output 3 enabled. • Bits 4...7: Not used. Association of HSC output Outx with physical output Qy is done by configuration.
F_Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
F_Preset	BOOL	Not used.

Input	Type	Description
F_Out	BYTE	Set bits to 1 to force corresponding physical outputs to 1 if associated with HSC by configuration. Takes priority over EN_Out. Only active when outputs configured in HSC editor: • Bit 0: Output 0 forced. • Bit 1: Output 1 forced. • Bit 2: Output 2 forced. • Bit 3: Output 3 forced. • Bits 4...7: Not used. Association of HSC output Outx with output terminal Qy is done by configuration.
ACK_Modulo	BOOL	Not used.
ACK_Preset	BOOL	Not used.
ACK_Cap	BOOL	Not used.
SuspendCompare	BOOL	TRUE = compare results are suspended: • Threshold, Reflex, and Out output bits of the function block maintain their last value. • Events are masked. **NOTE:** EN_Compare, EN_Reflex, and F_Out remain operational while SuspendCompare is set.

This table describes the output variables:

Outputs	Type	Comment
Run	BOOL	TRUE = Counter is activated. Edge to Edge: Set to 1 at rising edge detection, and reset to 0 at falling edge. Edge to Opposite: Set to 1 at first rising edge detection. It is only reset to 0 if the counter is disabled or an error is detected.
Valid	BOOL	TRUE = indicates that output values on the function block are valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See HSC_ERROR_TM3 *(see page 152)* enumeration.

Outputs	Type	Comment
Thresholds	BYTE	Bits set to 1 when CurrentValue ≥ Threshold *(see page 135)*: ● Bit 0: CurrentValue ≥ Threshold 0 ● Bit 1: CurrentValue ≥ Threshold 1 ● Bit 2: CurrentValue ≥ Threshold 2 ● Bit 3: CurrentValue ≥ Threshold 3 ● Bits 4...7: Not used Only active when EN_Compare is set.
Modulo_Flag	BOOL	Not used.
Preset_Flag	BOOL	Not used.
Cap_Flag	BOOL	Not used.
Reflex	BYTE	State of the reflex function: ● Bit 0: Reflex 0 ● Bit 1: Reflex 1 ● Bit 2: Reflex 2 ● Bit 3: Reflex 3 ● Bits 4...7: Not used
Out	BYTE	State of the physical outputs: ● Bit 0: Out0 ● Bit 1: Out1 ● Bit 2: Out2 ● Bit 3: Out3 ● Bits 4...7: Not used Association of HSC output Outx with physical output Qy is done by configuration.
CapturedValue	DINT	Not used.
CurrentValue	DINT	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table below can be read or modified by using the HSCGetParam_TM3 *(see page 171)* or HSCSetParam_TM3 *(see page 169)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start of the controller *(see Modicon M262 Logic/Motion Controller, Programming Guide)*.

Adjustable Parameters

This table provides the list of parameters from the HSC_PARAMETER_TYPE_TM3 *(see page 155)* which can be read or modified while the program is running:

Parameter	Description
HSC_PRESET	To get or set the Preset value of an HSC
HSC_TIMEBASE	To get or set the Resolution value of the HSC.
HSC_THRESHOLD0	To get or set the Threshold 0 value of an HSC
HSC_THRESHOLD1	To get or set the Threshold 1 value of an HSC
HSC_THRESHOLD2	To get or set the Threshold 2 value of an HSC
HSC_THRESHOLD3	To get or set the Threshold 3 value of an HSC
HSC_REFLEX0	To get or set output 0 reflex mode of an HSC function. The five least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 => TH0
HSC_REFLEX1	To get or set output 1 reflex mode of an HSC function. The five least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 => TH0
HSC_REFLEX2	To get or set output 2 reflex mode of an HSC function. The five least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 => TH0
HSC_REFLEX3	To get or set output 3 reflex mode of an HSC function. The five least significant bits indicate the reflex mode: 0x8 => TH3, 0x4 => TH2, 0x2 => TH1, 0x1 => TH0

Part VIII
Optional Functions

Overview

This part provides information on optional functions for HSC.

What Is in This Part?

This part contains the following chapters:

Chapter 17
Preset and Enable Functions

Overview

This chapter provides information on preset and enable functions for an HSC.

What Is in This Chapter?

This chapter contains the following topics:

Preset Function

Overview

The preset function is used to set/reset the counter operation.

The preset function authorizes counting function, synchronization, and start in the following counting modes:

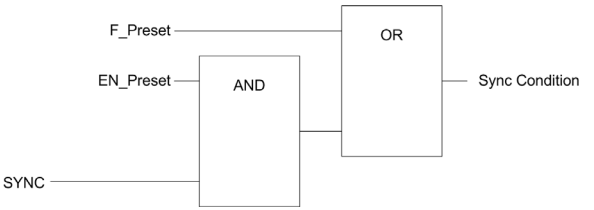
- **One shot** counter: preset and start the counter
- **Free-large** counter: preset and start the counter
- **Modulo-loop** counter: reset and start the counter
- **Event counting**: restart the internal time base at the beginning

NOTE: Sync condition for a **Simple** HSC type corresponds to the function block input `Sync`.

Description

This function is used to synchronize the counter depending on the status and the configuration of the optional SYNC physical input and the function block inputs `F_Preset` and `EN_Preset`.

This diagram illustrates the Sync conditions of the HSC:



EN_Preset input of the HSC function block
F_Preset input of the HSC function block
SYNC physical input SYNC

The function block output `Preset_Flag` is set 1 when the Sync Condition is reached.

Either of the following events trigger the capturing of the Sync Condition:

- Rising edge of the `F_Preset` input
- Rising edge, falling edge, or rising and falling edge, of the SYNC physical input (if the SYNC input is configured, and the `EN_Preset` input is TRUE).

Configuration

This procedure describes how to configure a preset function:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**.
2	Set the value of the **Counting function** parameter to **HSC Main Single Phase** or **HSC Main Dual Phase**.
3	Select the value of the **Control inputs → SYNC input → Location** parameter.
4	Select the value of the **Control inputs → SYNC input → Filter** parameter.
5	Select the value of the **Control inputs → SYNC input → Preset condition** parameter to specify the transition type of the SYNC physical input: • **SYNC Rising**. Rising edge of the SYNC input • **SYNC Falling**. Falling edge of the SYNC input • **SYNC Both**. Both edges of the SYNC input

Enable: Authorize Counting Operation

Overview

The enable function is used to authorize the counting operation.

The enable function is available in the following HSC modes:

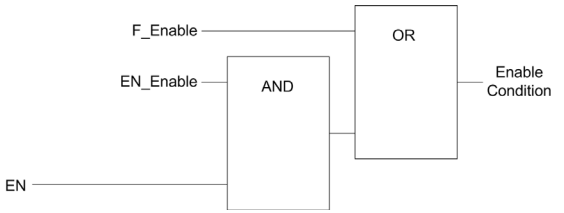
- HSC Main Single Phase (One-shot)
- HSC Main Single Phase (Modulo Loop)
- HSC Main Dual Phase (Modulo Loop)
- HSC Main Dual Phase (Free Large)
- Frequency Meter
- Period Meter

It is not available in Event Counting mode.

Description

This function is used to authorize changes to the counter value depending on the status of the optional `EN` physical input and the function block inputs `F_Enable` and `EN_Enable`.

The following diagram illustrates the enable conditions:



EN_Enable input of the HSC function block
F_Enable input of the HSC function block
EN physical input Enable

As long as the function is not enabled, the counting pulses are ignored.

NOTE: Enable condition for a **Simple** type corresponds to the function block input `Enable`.

Configuration

This procedure describes how to configure an Enable function:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**.
2	Select the **Counters** tab.
3	Select a **Counting function** that supports the Enable function: • **HSC Main Single Phase** (**One-shot** or **Modulo-loop**) • **HSC Main Dual Phase** (**Modulo-loop** or **Free-large**) • **Frequency Meter** • **Period Meter**
4	Set the value of the **Control inputs → EN input → Location** parameter.
5	Select the value of the **Control inputs → EN input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.

Chapter 18
Capture Function

Overview

This chapter provides information on capture function for HSC.

What Is in This Chapter?

This chapter contains the following topics:

Capture Principle with a Main Type

Overview

The capture function stores the counter value when an external input signal is detected.

The capture function is available in **Main** type with the following modes:

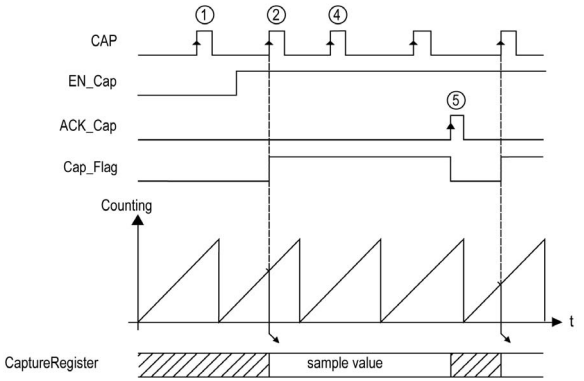
- One-shot *(see page 41)*
- Modulo-loop *(see page 61)*
- Free-large *(see page 79)*

To use this function:

- configure the optional Capture input **CAP**
- use the `CapturedValue` parameter of the HSCMain_TM3 *(see page 164)* function block to retrieve the captured value in your application.

Principle of a Capture

This graphic illustrates how the capture works in **Modulo-loop** mode:



Stage	Action
1	When `EN_Cap` = 0, the function is not operational.
2	When `EN_Cap` = 1, the edge on **CAP** captures the counter value, puts it into the Capture register, and triggers the rising edge of `Cap_Flag`.
3	Get the stored value using the `CapturedValue` parameter of the HSCMain_TM3 *(see page 164)* function block.
4	While `Cap_Flag` = 1, any new edge on the physical input CAP is ignored.
5	The rising edge of `HSCMain_TM3` *(see page 164)* function block input `ACK_Cap` triggers the falling edge `Cap_Flag` output. A new capture is authorized.

Configuration of the Capture on a Main Type

Configuration Procedure

Follow this procedure to configure the capture function on a **Main** type:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**.
2	Set the value of the **Counting function** parameter to **HSC Main Single Phase** or **HSC Main Dual Phase**.
3	Select a value for the **Capture → CAP input → Location**.
4	Select a value for the **Capture → CAP input → Filter** parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table *(see page 148)*.
5	Select a triggering mode for the **Capture → Mode** parameter: • Preset *(see page 126)* (default value) **NOTE:** The capture must be done before the preset action. • **CAP Rising** • **CAP Falling** • **CAP Both**

Chapter 19
Comparison Function

Overview

This chapter provides information on the comparison function for the HSC.

What Is in This Chapter?

This chapter contains the following topics:

Comparison Principle with a Main type

Overview

The compare block with the **Main** type manages thresholds, reflex outputs and events in the following modes:

- One-shot *(see page 35)*
- Modulo-loop *(see page 49)*
- Free-Large *(see page 71)*
- Period Meter *(see page 107)*

Comparison is configured in the Configuration screen by activating at least one threshold.

Comparison can be used to trigger:

- a programming action on thresholds *(see page 138)*
- an event on a threshold associated with an external task *(see page 137)*
  NOTE: This option is only available for TM3XF• expansion modules, which support external events.
- reflex outputs *(see page 139)*.

Principle of a Comparison

The **Main** type can manage up to four thresholds.

A threshold is a configured value that is compared to the counting value. Thresholds are used to define up to five zones or to react to a value crossing the threshold value.

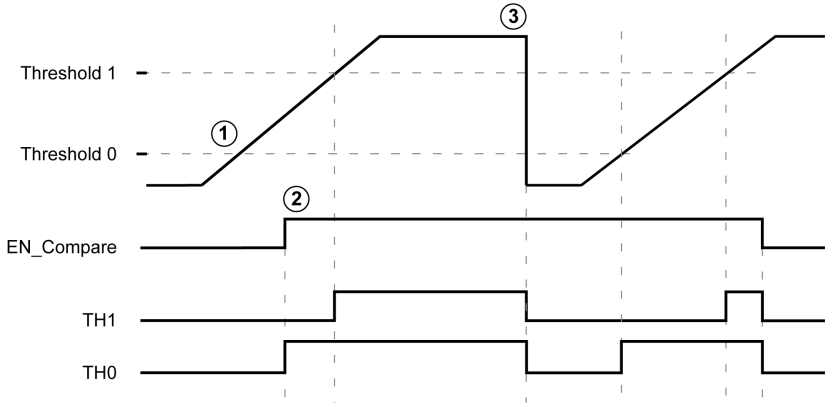
Threshold values are defined in the configuration window and can also be adjusted in the application program by using the `HSCSetParam_TM3` *(see page 169)* function block.

If Thresholdx (x= 0, 1, 2, 3) is configured and comparison is enabled (`EN_Compare` = 1), output pin THx of the `HSCMain_TM3` function block is:

- set when counter value >= Thresholdx
- reset when counter value < Thresholdx

NOTE: When `EN_Compare` is set to 0 on `HSCMain_TM3` function block, comparison functions are disabled, including external tasks triggered by a threshold event and Reflex outputs.

The following example for Modulo loop with two thresholds shows comparison in the `HSCMain_TM3` function block:



Stage	Action
1	When `EN_Compare = 0`, the function is not operational.
2	When `EN_Compare = 1` as the counter value is already over `Threshold 0`, `TH0` is set to 1.
3	The counter is reset, due to a synchronization condition for example.

Configuring Event Triggering in HSC Main Single or Dual Phase

Configuring an event on threshold crossing allows to trigger an external task. You can choose to trigger an event when a configured threshold is crossed as follows:

- **Upward Cross**. The event is triggered when the measured value goes above the threshold value.
- **Downward Cross**. The event is triggered when the measured value goes below the threshold value.
- **Both Cross**. The event is triggered when the measured value goes above the threshold value and when the measured value goes below the threshold value.

Configuring Event Triggering in Period Meter Mode

Configuring an event allows to trigger an external task. You can choose to trigger an event as follows:

- **Below threshold value**. The event is triggered when the measured value is lower than the threshold value.
- **Above threshold value**. The event is triggered when the measured value is higher than the threshold value.
- **Between threshold values**. The event is triggered when the measured value is between two threshold values.

Threshold Behavior

Using thresholds comparison status available in the task context (`TH0` to `TH3` output pins of the function block) is suitable for an application tolerant of the inherent lag of cycle times and asynchronism of communications, especially when using the modules over a field bus in distributed architectures.

Configuring Reflex Output

Follow this procedure to configure reflex outputs:

Step	Action
1	In **Compare → Thresholds → Number of thresholds** select a number of thresholds. **Result: Threshold values** and **Reflex Outputs** are displayed.
2	Enter the value in the **value** field of each threshold value. **NOTE:** EcoStruxure Machine Expert follow this rule to configure the threshold values and adapt them if necessary: **TH0 < TH1 < TH2 < TH3 < TH4**. **NOTE:** For **HSC Main** functions, you can set a higher value for thresholds than defined in **Preset** field.
3	Configure the **Reflex Outputs**.

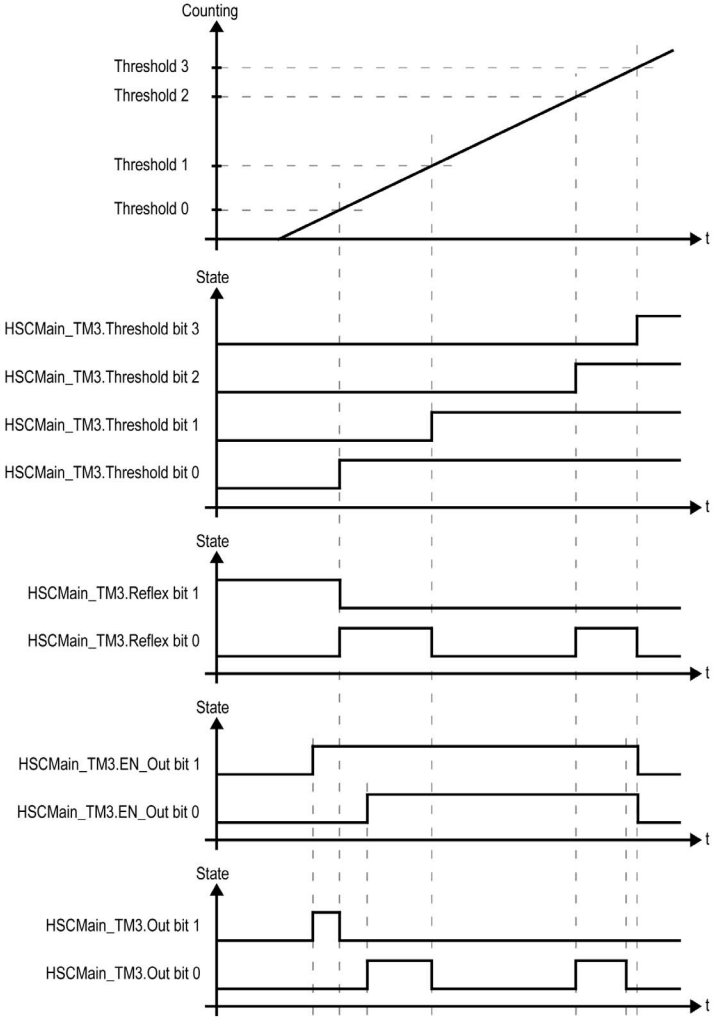
Reflex Output Behavior

Configuring reflex outputs allows to trigger physical reflex outputs.

These outputs are not controlled in the task context, reducing the reaction time to a minimum. This is convenient for operations that need fast execution.

Outputs used by the High Speed Counter can only be accessed through the function block. They cannot be read or written directly within the application.

Example of the reflex outputs triggered by threshold:



NOTE: The state of the reflex outputs depends on the configuration.

Changing the Threshold Values

Care must be exercised when threshold compares are active to avoid unintended or unexpected results from the outputs or from sudden Event task execution. If the compare function is disabled, threshold values can be modified freely. However, if the compare function is enabled, suspend at least the threshold compare function while modifying the threshold values.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not change the Threshold values without using the `SuspendCompare` input if `EN_Compare` is equal to 1.
- Verify that `TH0` is less than `TH1`, that `TH1` is less than `TH2`, and that `TH2` is less than `TH3` before reactivating the threshold compare function.

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

While `EN_Compare` = 1, the comparison is active, and it is necessary to follow this procedure to apply changes to threshold values:

Step	Action
1	Set `SuspendCompare` to 1. The comparison is frozen at the counter value: ● The `Thresholds`, `Reflex`, and `Out` output bits of the function block maintain their last value. ● Physical outputs 0, 1 maintain their last value ● Events are masked **NOTE:** `EN_Compare`, `EN_Out`, and `F_Out` remain operational while `SuspendCompare` is set.
2	Modify the threshold values as needed using the `HSCSetParam_TM3` *(see page 169)* function block.
3	Set `SuspendCompare` to 0. The new threshold values are applied and the comparison is resumed.

Configuration of the Comparison on a Main Type

Configuration Procedure for HSC Main Single or Dual Phase

Follow this procedure to configure the comparison function on a **HSC Main Single Phase** or **HSC Main Dual Phase** type:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**.
2	Set the value of the **Counting function** parameter to **HSC Main Single Phase** or **HSC Main Dual Phase**.
3	In the **Number of thresholds** parameter, select the number of thresholds to use.
4	Set the value of each threshold. **NOTE:** Follow this rule to configure the threshold values: TH0 < TH1 < TH2 < TH3
5	**NOTE:** This option is only available for TM3XF• expansion modules, which support external events. Optionally, define event conditions for the thresholds: 1. Configure external events associated with tasks. 2. In **Events → Threshold x → Threshold x Event**, select a trigger type (**Upward Cross**, **Downward Cross**, **Both Cross**) 3. In **Threshold x Event Name**, enter an external event name. To use the event, this external event name must be referenced in a task.

Configuration Procedure for Period Meter

Follow this procedure to configure the comparison function on a **Period Meter Main** type:

Step	Action
1	In the **Devices tree**, double-click **MyController → IO_Bus → Module_x → Counters**.
2	Set the value of the **Counting function** parameter to **Period Meter**.
3	In the **Number of thresholds** parameter, select the number of thresholds to use.
4	Set the value of each threshold. **NOTE:** Follow this rule to configure the threshold values: TH0 < TH1 < TH2 < TH3
5	Optionally, in the **Reflex outputs** section, select the physical outputs to use for reflex output signals. The outputs can be activated when the value of the counter is within defined intervals (see below).

Step	Action
6	Depending on the number of thresholds activated, predefined **Intervals** are established between: ● The minimum value and TH0 ● TH0 and TH1 ● TH1 to TH2 ● TH2 to TH3 ● TH3 and the maximum value. Optionally, for each interval set the **Reflex x Output** value to **TRUE**. The reflex output is then set if the counter value is greater than or equal to the lower threshold value and less than the upper threshold value.
7	**NOTE:** This option is only available for TM3XF• expansion modules, which support external events. Optionally, define event conditions for the thresholds: **1.** Configure external events associated with tasks. **2.** In **Events → Event x → Event x**, select a trigger type: ❍ **Below TH0** ❍ **Above TH0** ❍ **Between TH0 and TH1** ❍ **Below TH1** ❍ **Above TH1** ❍ **Between TH1 and TH2** ❍ **Below TH2** ❍ **Above TH2** ❍ **Between TH2 and TH3** ❍ **Above TH3** An external event is triggered whenever the counter value falls within the selected range. **3.** In **Event x Name**, enter an external event name. To use the event, this external event name must be referenced in a task.

External Event Configuration

Overview

On the TM3XFHSC202 module, external events can be configured to activate external tasks.

When an external event is configured on a Counting function:

- An external event is created and its default name displayed in the **Counters** configuration window. The default name can be edited.
- The external event name is available when configuring an external task.

Procedure

The following procedure describes how to configure an external event to activate a task:

Step	Action
1	In the **Applications tree** tab, add a task.
2	Double-click the **Task** node to associate it with to an external event.
3	In the **Type** dropdown menu on the Configuration window, select **External**.
4	In the **External event** drop down menu, select the event to associate to the task (see the list below). **NOTE:** The task must be configured when an event is declared.

External Events

This table provides a description of the possible external events to associate to a task:

Event Name	Description
I*x*, *x*=0...9	Task is activated when the input Ix signal detects a rising edge, falling edge, or both edges. The type of signal detected is configured on the Configuration window.
*instancename*_TH*x*	Configurable in the task is activated when **Threshold Value x** is crossed. Task activation can be triggered when the High Speed Counter is configured as counting up, counting down, or both.
*instancename*_TH*x*_TH*y*	Task is activated when counter value is between Threshold *x* and Threshold *y*.
*instancename*_Below_TH*x*	Task is activated when counter value is below Threshold *x*.
*instancename*_Above_TH*y*	Task is activated when counter value is above Threshold *x*.
*instancename*_STOP	Only available in HSC Main Single Phase *(see page 43)* mode. Task is activated when counter value is set to 0.
*instancename*_PERIOD	Only available in Period Meter Edge to Edge *(see page 115)* and Period Meter Edge to Opposite *(see page 116)* modes. Task is activated at the end of a configured period.

Appendices

Overview

This appendix extracts parts of the programming guide for technical understanding of the library documentation.

What Is in This Appendix?

The appendix contains the following chapters:

Appendix A
General Information

What Is in This Chapter?

This chapter contains the following topics:

Dedicated Features

Bounce Filter

This table shows the maximum counter frequencies determined by the filtering values used to reduce the bounce effect on the input:

Input	Bounce Filter Value (ms)	Maximum Counter Frequency Expert
A B	0.000	200 kHz
	0.001	200 kHz
	0.002	200 kHz
	0.005	100 kHz
	0.01	50 kHz
	0.05	25 kHz
	0.08	6.25 kHz
	0.5	1 kHz
	1	500 Hz
	4	125 Hz
	12	40 Hz
A is the counting input of the counter. B is the counting input of the dual phase counter.		

Dedicated Outputs

Outputs used by the high speed expert functions can only be accessed through the function block. They cannot be read or written directly within the application.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use the same function block instance in different program tasks.
- Do not modify or otherwise change the function block reference (AXIS) while the function block is executing.

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

General Information on Administrative Function Block Management

Overview

The high speed counter (HSC) library of the TM3 expert modules includes the GetParam_TM3 *(see page 171)* and SetParam_TM3 *(see page 169)* administrative function blocks.

The following provides general information on managing common input and output variables.

Management of Input Variables

At the Execute input rising edge, the function block starts.

Any further modifications of the input variables are not taken into account.

Following the IEC 61131-3 standards, if any variable input to a function block is missing, that is, left open or unconnected, then the value from the previous invocation of the instance of the function block will be used. In the first invocation, the initial, configured value is applied in this case. Therefore, it is best that a function block always has known values attributed to its inputs to help avoid difficulties in debugging your program. For HSC function blocks, it is best to use the instance only once, and preferably use the instance in the main task.

Management of Output Variables

The Done output is mutually exclusive with the Busy and Error outputs: only one of them can be TRUE on one function block. If the Execute input is TRUE, one of these outputs is TRUE.

At the rising edge of the Execute input, the Busy output is set. This Busy output remains set during the function block execution, and is reset at the rising edge of one of the other outputs (Done or Error).

The Done output is set when the function block execution has been completed successfully.

When a function block execution ends due to a detected error, the Error output is set and the detected error number is given through the ErrorID output.

The Done, Error, and ErrorID outputs are reset with the falling edge of Execute. If the Execute input is reset before execution is finished, then the outputs are set for one task cycle at the execution ending.

When an instance of a function block receives a new Execute before it is finished, the function block does not return any feedback, such as Done, for the previous action.

Handling a Detected Error

All HSC function blocks have 2 outputs that can report a detected error during the execution of the function block:

- Error = TRUE when an error is detected.
- ErrorID When Error = TRUE, returns an HSC_ERROR_TM3 *(see page 152)* detected error ID.

Appendix B
Data Types

Overview

This chapter describes the data types of the HSC Library.

What Is in This Chapter?

This chapter contains the following topics:

HSC_ERROR_TM3: HSC Variable Detected Error Type Block

Enumerated Type Description

The enumeration data type ENUM contains the different types of detected error with the following values:

Enumerator	Value	Description
HSC_NO_ERROR	00 hex	No error detected.
HSC_UNKNOWN_REF	01 hex	The HSC reference is incorrect or not configured.
HSC_UNKNOWN_PARAMETER	02 hex	The parameter reference is incorrect. See HSC_PARAMETER_TYPE_TM3 *(see page 155)* for valid parameters.
HSC_INVALID_PARAMETER	03 hex	The value of the parameter is incorrect. For example, `Preset Value` is less than TH1 or TH0.
HSC_COM_ERROR	04 hex	Communication error was detected with the HSC module.
HSC_OPTIONAL_MODULE_NOT_THERE	05 hex	An optional module is not in the current configuration.
HSC_PERIODMETER_TIMEOUT_REACHED	06 hex	A timeout specified for period measurement expired.
HSC_SHORTCUT_GROUP0_DETECTED	07 hex	Output shortcut group 0 detected.
HSC_SHORTCUT_GROUP1_DETECTED	08 hex	Output shortcut group 1 detected.
HSC_SHORTCUT_GROUP2_DETECTED	09 hex	Not used.
HSC_SHORTCUT_GROUP3_DETECTED	0A hex	Not used.
HSC_24VIO_ERROR	0B hex	24 V I/O signal not detected.
HSC_EVENT_OVERFLOW_ERROR	0C hex	Events overflow (a maximum of 8 events per millisecond is allowed).

HSC_EVENT_TIMEBASE_TYPE_TM3: Type for Event Time Base Variable

Enumerated Type Description

This enumeration data type ENUM is used to configure a **Main** type single phase in **Event Counting** mode.

The enumeration contains the different time base values allowed for use with the HSCSetParam_TM3 *(see page 169)* and HSCGetParam_TM3 *(see page 171)* function blocks:

Name	Value
HSC_EVENT_100ms	0
HSC_EVENT_1s	1
HSC_EVENT_10s	2
HSC_EVENT_60s	3

HSC_FREQMETER_TIMEBASE_TYPE_TM3: Type for Frequency Meter Time Base Variable

Enumerated Type Description

This enumeration data type ENUM is used in Frequency Meter mode.

The enumeration contains the different time base values allowed for use with an HSC function block:

Name	Value
HSC_FREQMETER_10ms	10
HSC_FREQMETER_100ms	100
HSC_FREQMETER_1000ms	1000

HSC_PARAMETER_TYPE_TM3: Type for Parameters to Get or to Set on HSC Function Blocks

Enumerated Type Description

The enumeration data type ENUM contains the following values:

Enumerator	Value	Description
HSC_PRESET	00 hex	To get or set the Preset value of an HSC function.
HSC_MODULO	01 hex	To get or set the Modulo value of an HSC function.
HSC_TIMEBASE	03 hex	To get or set the Timebase value for Event Counting and Frequency Meter modes, or the Resolution value for Period Meter mode.
HSC_THRESHOLD0	06 hex	To get or set the Threshold 0 value.
HSC_THRESHOLD1	07 hex	To get or set the Threshold 1 value.
HSC_THRESHOLD2	08 hex	To get or set the Threshold 2 value.
HSC_THRESHOLD3	09 hex	To get or set the Threshold 3 value.
HSC_REFLEX0	0A hex	To get or set output 0 reflex mode for HSC Main and Period Meter modes. The four least significant bits correspond to a reflex mode: ● 0x8 => TH3 ● 0x4 => TH2 ● 0x2 => TH1 ● 0x1 => TH0
HSC_REFLEX1	0B hex	To get or set output 1 reflex mode for HSC Main and Period Meter modes The four least significant bits correspond to a reflex mode: ● 0x8 => TH3 ● 0x4 => TH2 ● 0x2 => TH1 ● 0x1 => TH0

Enumerator	Value	Description
HSC_REFLEX2	0C hex	To get or set output 2 reflex mode for HSC Main and Period Meter modes. The four least significant bits correspond to a reflex mode: ● 0x8 => TH3 ● 0x4 => TH2 ● 0x2 => TH1 ● 0x1 => TH0
HSC_REFLEX3	0D hex	To get or set output 3 reflex mode for HSC Main and Period Meter modes The four least significant bits correspond to a reflex mode: ● 0x8 => TH3 ● 0x4 => TH2 ● 0x2 => TH1 ● 0x1 => TH0

HSC_PERIODMETER_RESOLUTION_TYPE_TM3: Type for Period Meter Time Base Variable

Enumerated Type Description

This enumeration data type ENUM is used in Period Meter mode.

The enumeration contains the different time base values allowed for use with an HSC function block:

Name	Value
HSC_PERIODMETER_100ns	FFFFFFFF hex (-1 decimal)=100 ns
HSC_PERIODMETER_1us	00 hex (0 decimal)=1 µs
HSC_PERIODMETER_100us	01 hex (1 decimal)=100 µs
HSC_PERIODMETER_1000us	02 hex (2 decimal)=1000 µs

Appendix C
Functions

GetExternalEventValue: Get Current Value of an External Event

Function Description

Use this function to get the value associated with an external event task.

NOTE: The function must be called from within an external event task.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
pValue	POINTER TO DINT	Address of the variable where the value is copied if the function returns EXTEVT_VAL_OK.

This table describes the output variables:

Outputs	Type	Comment
GetExternalEventValue	EXTEVT_VAL_RES	Returns one of the following values: • EXTEVT_VAL_OK: Valid value • EXTEVT_VAL_FAILED: Unable to obtain value • EXTEVT_VAL_NOT_IN_EXT_EVT_TASK: Function was not called from within an external event task • EXTEVT_VAL_NOT_AVAILABLE: No value available for this external task

Appendix D
Function Blocks

Overview

What Is in This Chapter?

This chapter contains the following topics:

HSCSimple_TM3: Control a Simple Type Counter for TM3

Function Block Description

This function block controls a **Simple** type counter with the following reduced functions:

- one-channel counting
- no threshold
- no capture
- no reflex

The `HSCSimple` function block is mandatory when using a **Simple** counter type.

The function block instance name must match the name defined by configuration. Hardware related information managed by this function block is synchronized with the MAST task cycle.

⚠ WARNING

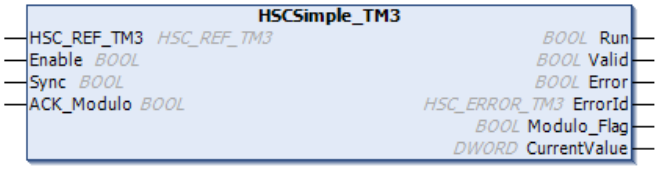
UNINTENDED OUTPUT VALUES

- Only use the Function Block instance in the MAST task.
- Do not use the same Function Block instance in a different task.

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

NOTE: Forcing the logical output values of the FB is allowed by EcoStruxure Machine Expert but it will have no impact on hardware related outputs if the function is active (executing).

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC instance.
Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
Sync	BOOL	One shot counter: On rising edge, loads the preset of the counter. Modulo loop counter: On rising edge, resets and initializes the counter.
ACK_Modulo	BOOL	One shot counter: Not used. Modulo loop mode: On rising edge, resets the modulo flag Modulo_Flag.

This table describes the output variables:

Outputs	Type	Comment
Validity	BOOL	TRUE = indicates that the output values on the function block are valid.
Run	BOOL	TRUE = counter is activated. In One-shot mode, switches to 0 when CurrentValue reaches 0. A rising edge on Sync is needed to restart the counter. In Modulo loop mode, TRUE, indicates that the counter is activated.
Valid	BOOL	TRUE = indicates that output values on the function block are valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See HSC_ERROR_TM3 *(see page 152)* enumeration.
Modulo_Flag	BOOL	One shot counter: Not relevant. Module loop mode: Set to TRUE when the counter rolls over the modulo value.
CurrentValue	DWORD	The value of the counter.

HSCMain_TM3: Control a Main Type Counter for TM3

Function Block Description

This function block controls a **Main** type counter with the following functions:

- up/down counting
- frequency meter
- thresholds
- events
- period meter
- dual phase

The HSC Main function block is mandatory when using **Main** counter.

The function block instance name must match the name defined by configuration. Hardware related information managed by this function block is synchronized with the MAST task cycle.

⚠ WARNING

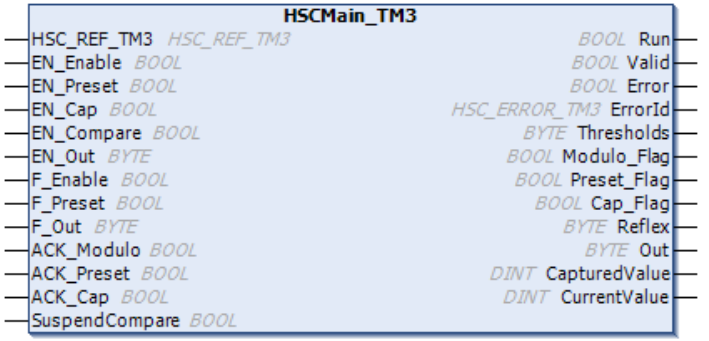
UNINTENDED OUTPUT VALUES

- Only use the Function Block instance in the MAST task.
- Do not use the same Function Block instance in a different task.

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

NOTE: Forcing the logical output values of the FB is allowed by EcoStruxure Machine Expert but it will have no impact on hardware related outputs if the function is active (executing).

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Input	Type	Description
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC instance.
EN_Enable	BOOL	TRUE = authorizes enabling of the counter using the Enable input.
EN_Preset	BOOL	TRUE = authorizes counter synchronization and start using the Sync input.
EN_Cap	BOOL	TRUE = enables the Capture input (if configured in **One shot**, **Modulo loop**, **Free large** modes).
EN_Compare	BOOL	TRUE = enables the comparator operation (using Thresholds 0, 1, 2, 3): • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1, Reflex2, Reflex3 output bits) • events (to trigger external tasks on threshold crossing) **NOTE:** This option is only available for TM3XF• expansion modules, which support external events.
EN_Out	BYTE	Set bits to 1 to enable the corresponding physical outputs to echo the configured function value (Reflex or Stop) as a result of the comparison function. Only active when outputs configured in HSC editor: • Bit 0: Output 0 enabled. • Bit 1: Output 1 enabled. • Bit 2: Output 2 enabled. • Bit 3: Output 3 enabled. • Bits 4...7: Not used. Association of HSC output Outx with physical output Qy is done by configuration (**One-shot**, **Modulo loop**, **Free large**, **Period Meter Edge to Edge** and **Period Meter Edge to Opposite** modes).
F_Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
F_Preset	BOOL	On rising edge, authorizes counting function synchronization and start in the following counting modes: **One-shot counter:** to preset and start the counter **Modulo loop counter:** to reset and start the counter **Free large counter:** to preset and start the counter **Event counter:** to restart the internal time base at the beginning **Frequency meter:** to restart the internal timer relative to the time base. The counter value is not preset.

Input	Type	Description
F_Out	BYTE	Set bits to 1 to force the corresponding physical outputs to 1 if associated with HSC by configuration. Takes priority over EN_Out. Only active when outputs configured in HSC editor: • Bit 0: Output 0 forced. • Bit 1: Output 1 forced. • Bit 2: Output 2 forced. • Bit 3: Output 3 forced. • Bits 4...7: Not used. Association of HSC output Outx with physical output Qy is done by configuration (**One-shot**, **Modulo loop**, **Free large**, **Period Meter Edge to Edge** and **Period Meter Edge to Opposite** modes).
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag (**Modulo loop** and **Free large** modes). No effect in **One-shot** mode.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag. Not applicable in **Period Meter** mode.
ACK_Cap	BOOL	On rising edge, resets the Cap_Flag (**One-shot**, **Modulo loop**, **Free large** modes).
SuspendCompare	BOOL	TRUE = compare results are suspended: • Events are masked. **NOTE:** EN_Compare, EN_Reflex, F_Out remain operational while SuspendCompare is set. Not applicable in **Event Counting** and **Frequency Meter** modes.

This table describes the output variables:

Outputs	Type	Comment
Run	BOOL	TRUE = counter is activated. **One shot** mode: The Run bit is set to 0 when counter value reaches 0. A preset is needed to restart the counter. **Period Meter Edge to Edge** mode: The Run bit is set to 1 at rising edge detection, and reset to 0 at falling edge. **Period Meter Edge to Opposite** mode: The Run bit is set to 1 at first rising edge detection. It is only reset to 0 if the counter is disabled or an error is detected.
Valid	BOOL	Set to TRUE when CurrentValue is valid.
Error	BOOL	TRUE = indicates that an error was detected.
ErrorId	HSC_ERROR_TM3	Indicates the value of the error detected. See HSC_ERROR_TM3 *(see page 152)* enumeration.

Outputs	Type	Comment
Thresholds	BYTE	Bits set to 1 when CurrentValue ≥ Threshold *(see page 135)* for corresponding threshold: • Bit 0: CurrentValue ≥ Threshold 0 • Bit 1: CurrentValue ≥ Threshold 1 • Bit 2: CurrentValue ≥ Threshold 2 • Bit 3: CurrentValue ≥ Threshold 3 • Bits 4...7: Not used Not applicable in **Event Counting** and **Frequency Meter** modes.
Modulo_Flag	BOOL	Set to TRUE when the counter rolls over its limits in the following modes: • **Modulo loop counter**: when the counter rolls over to the modulo or 0 • **Free large** counter: when the counter roll overs its limits Applicable only in **HSC Main Single Phase** and **HSC Main Dual Phase** modes.
Preset_Flag	BOOL	Set to TRUE by the synchronization of: • **One-shot** counter: when the counter presets and starts • **Modulo loop** counter: when the counter resets • **Free large** counter: when the counter presets • **Event** counter: when the internal timer relative to the time base restarts • **Frequency meter**: when the internal timer relative to the time base restarts
Cap_Flag	BOOL	TRUE = indicates that a value has been latched in the capture register. This flag must be reset before a new capture can occur. Not applicable in **Event Counting**, **Period Meter**, and **Frequency Meter** modes.
Reflex	BYTE	State of the reflex function: • Bit 0: Reflex 0 • Bit 1: Reflex 1 • Bit 2: Reflex 2 • Bit 3: Reflex 3 • Bits 4...7: Not used Not applicable in **Event Counting** and **Frequency Meter** modes.

Outputs	Type	Comment
Out	BYTE	State of the physical outputs: ● Bit 0: Output 0 ● Bit 1: Output 1 ● Bit 2: Output 2 ● Bit 3: Output 3 ● Bits 4...7: Not used Only active when the outputs are configured in the **Counters** configuration tab. Association of HSC output Outx with physical output Qy is done by configuration. Not applicable in **Event Counting** and **Frequency Meter** modes.
CapturedValue	DINT	Set to TRUE when CurrentValue is valid. Not used in **Period Meter** mode.
CurrentValue	DINT	The value of the counter.

HSCSetParam_TM3: Adjust Parameters of a HSC

Function Block Description

This administrative function block modifies the value of a parameter of an HSC.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
HSC_REF_TM3	HSC_REF_TM3	Reference to the HSC function block. Must not be changed during block execution.
Execute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
Param	HSC_PARAMETER_TYPE_TM3 *(see page 155)*	Parameter to set.
ParamValue	DINT	Parameter value to write. If the requested parameter is the Timebase of an HSCMain Single, use the HSC_EVENT_TIMEBASE_TYPE_TM3 *(see page 153)* enumeration to set the value. If the requested parameter is the Timebase of a Frequency Meter, use the HSC_FREQMETER_TIMEBASE_TYPE_TM3 *(see page 154)* enumeration to set the value. If the requested parameter is the Resolution of a Period Meter, use the HSC_PERIODMETER_RESOLUTION_TYPE_TM3 *(see page 157)* enumeration to set the value.

This table describes the output variables:

Outputs	Type	Comment
Done	BOOL	TRUE = indicates that the parameter was successfully written. Function block execution is finished.
Busy	BOOL	TRUE = indicates that the function block execution is in progress.
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrorID	HSC_ERROR_TM3 *(see page 152)*	Indicates the value of the error detected.

NOTE: For more information about Done, Busy, and Execute pins, refer to General Information on Administrative Function Block Management *(see page 149)*.

Adding the HSCSetParam_TM3 Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3 HSC → Administrative → HSCSetParam_TM3** in the list, drag-and-drop the item onto the **POU** window.
2	Set the **HSC_REF_TM3** input to the instance name of the HSC.

HSCGetParam_TM3: Returns Parameters of HSC

Function Block Description

This administrative function block returns a parameter value of an HSC.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 173)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
HSC_REF_TM3	HSC_REF_TM3	Reference of the HSC instance. Must not be changed during function block execution.
Execute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
Param	HSC_PARAMETER_TYPE_TM3 *(see page 155)*	Parameter to read.

This table describes the output variables:

Outputs	Type	Comment
Done	BOOL	TRUE = indicates that ParamValue is valid. Function block execution is finished.
Busy	BOOL	TRUE = indicates that the function block execution is in progress.
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrorID	HSC_ERROR_TM3 *(see page 152)*	Indicates the value of the error detected.
ParamValue	DINT	Value of the parameter that has been read.

NOTE: For more information about the Execute, Done, and Busy pins, refer to General Information on Administrative Function Block Management *(see page 149)*.

Adding the HSCGetParam_TM3 Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Intern → IODrivers → TM3 HSC → Administrative → HSCGetParam_TM3** in the list, drag-and-drop the item onto the **POU** window.
2	Set the value of the **HSC_REF_TM3** input to the instance name of the HSC.

Appendix E
Function and Function Block Representation

Overview

Each function can be represented in the following languages:

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

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

What Is in This Chapter?

This chapter contains the following topics:

Differences Between a Function and a Function Block

Function

A function:

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

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

Function Block

A function block:

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

Examples: timers, counters

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

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

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

General Information

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

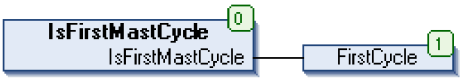
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

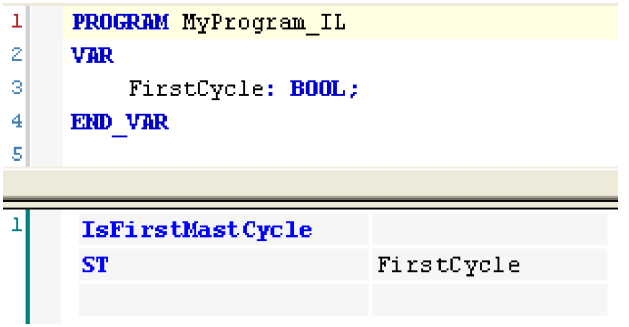
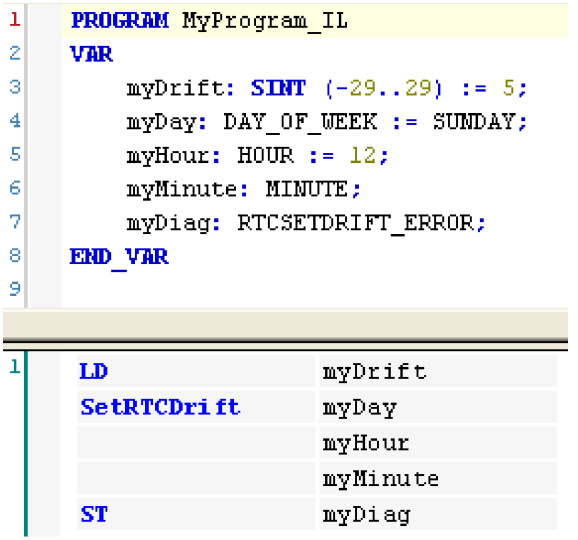
This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: ● type the name of the function in the operator column (left field), or ● use the **Input Assistant** to select the function (select **Insert Box** in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with `???` in the fields on the right. Replace the `???` with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

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

Function	Graphical Representation
without input parameter: `IsFirstMastCycle`	
with input parameters: `SetRTCDrift`	

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

Function	Representation in POU IL Editor
IL example of a function without input parameter: `IsFirstMastCycle`	
IL example of a function with input parameters: `SetRTCDrift`	

Using a Function Block in IL Language

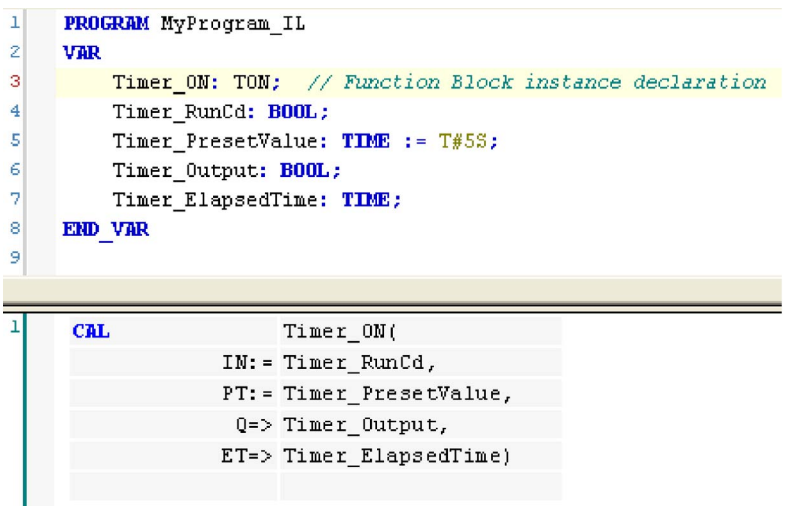
This procedure describes how to insert a function block in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs.
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a `CAL` instruction: ● Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in the context menu). ● Automatically, the `CAL` instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: ● Values to inputs are set by ":=". ● Values to outputs are set by "=>".
4	In the `CAL` right-side field, replace `???` with the instance name.
5	Replace other `???` with an appropriate variable or immediate value.

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

Function Block	Graphical Representation
`TON`	

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

Function Block	Representation in POU IL Editor
TON	

```
PROGRAM MyProgram_IL
VAR
    Timer_ON: TON;  // Function Block instance declaration
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
CAL          Timer_ON(
        IN:= Timer_RunCd,
        PT:= Timer_PresetValue,
         Q=> Timer_Output,
        ET=> Timer_ElapsedTime)
```

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

General Information

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

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

Using a Function in ST Language

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

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

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

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Using a Function Block in ST Language

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

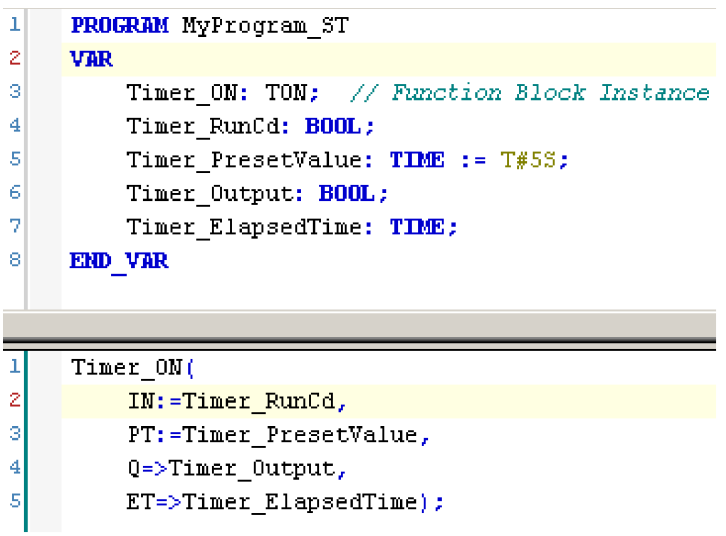
Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block

3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is:
	`FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,...`
	`Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

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

Function Block	Graphical Representation
`TON`	

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

Function Block	Representation in POU ST Editor
TON	

```
PROGRAM MyProgram_ST
VAR
    Timer_ON: TON;  // Function Block Instance
    Timer_RunCd: BOOL;
    Timer_PresetValue: TIME := T#5S;
    Timer_Output: BOOL;
    Timer_ElapsedTime: TIME;
END_VAR
```

```
Timer_ON(
    IN:=Timer_RunCd,
    PT:=Timer_PresetValue,
    Q=>Timer_Output,
    ET=>Timer_ElapsedTime);
```

Glossary

A

application

A program including configuration data, symbols, and documentation.

B

byte

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

C

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.

F

FB

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

function block diagram

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

I

ID

(*identifier/identification*)

IEC 61131-3

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

IL

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

INT

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

L

LD

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

N

node

An addressable device on a communication network.

P

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.

S

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.

V

variable

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

Index

M

P