

Modicon M258 Logic Controller

High Speed Counting
M258 Expert I/O Library Guide

04/2012

E100000000572.04

www.schneider-electric.com



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

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

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

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

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

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

© 2012 Schneider Electric. All rights reserved.

Table of Contents



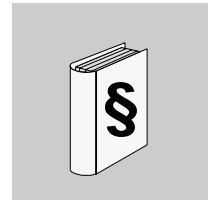
	Safety Information	7
	About the Book	9
Part I	High Speed Counter and Encoder Overview	13
Chapter 1	Introduction	15
	Expert I/O Overview	16
	Add an Expert function	19
	Embedded Expert I/O mapping	22
Chapter 2	High Speed Counter Overview	25
	Simple Type Overview	26
	Main Type Overview	27
	Choosing your HSC	28
Chapter 3	Encoder Overview	31
	Standard Encoder Overview	31
Part II	One-shot Mode	33
Chapter 4	One-shot Mode Principle	35
	One-shot Mode Principle Description	35
Chapter 5	One-shot With a Simple Type	37
	Synopsis Diagram	38
	Configuration of the Simple Type in One-shot Mode	39
	Programming the Simple Type	41
	Adjusting Parameters	43
Chapter 6	One-shot With a Main Type	45
	Synopsis Diagram	46
	Configuration of the Main Type in One-shot Mode	47
	Programming the Main Type	50
	Adjusting Parameters	53
Part III	Modulo-loop Mode	55
Chapter 7	Modulo-loop Principle	57
	Modulo-loop Mode Principle Description	57

Chapter 8	Modulo-loop with a Simple Type	59
	Synopsis Diagram	60
	Configuration of the Simple Type in Modulo-loop Mode	61
	Programming the Simple Type	63
	Adjusting Parameters.	65
Chapter 9	Modulo-loop With a Main Type	67
	Synopsis Diagram	68
	Configuration of the Main Type in Modulo-loop Mode	69
	Programming the Main Type	72
	Adjusting Parameters.	75
Part IV	Free-large Mode	77
Chapter 10	Free-large Mode Principle	79
	Free-large Mode Principle Description.	80
	Limits Management	83
Chapter 11	Free-large With a Main Type	85
	Synopsis Diagram	86
	Configuration of the Main Type in Free-large Mode	87
	Programming the Main Type	90
	Adjusting Parameters.	93
Part V	Event Counting Mode	95
Chapter 12	Event Counting Principle	97
	Event Counting Mode Principle Description.	97
Chapter 13	Event Counting With a Main Type	99
	Synopsis Diagram	100
	Configuration of the Main Type in Event Counting Mode	101
	Programming the Main Type	104
	Adjusting Parameters.	107
Part VI	Frequency Meter Mode	109
Chapter 14	Frequency Meter Principle	111
	Frequency Meter Mode Principle Description	111
Chapter 15	Frequency Meter Mode With a Main Type	113
	Synopsis Diagram	114
	Configuration of the Main Type in Frequency Meter Mode	115
	Programming the Main Type	117
Part VII	Period Meter Mode	121
Chapter 16	Period Meter Mode Principle	123
	Period Meter Mode Principle Description	123
Chapter 17	Period Meter With a Main Type	125
	Synopsis Diagram	126
	Configuration of the Main Type in Period Meter Mode	127
	Programming the Main Type	130

Part VIII	Encoder	133
Chapter 18	Incremental Mode With an Encoder	135
	Incremental Mode Principle Description	136
	Synopsis Diagram	140
	Configuration of the Standard Encoder on an Expert I/O Module	141
	Programming the Standard Encoder	146
	Adjusting Parameters	149
Part IX	Optional Functions	151
Chapter 19	Comparison Function	153
19.1	Comparison with a Main Type	153
	Comparison Principle with a Main type or an Encoder	154
	Configuration of the Comparison on a Main Type or an Encoder	158
	External Event Configuration	159
Chapter 20	Capture Function	161
20.1	Capture with a Main Type	162
	Capture Principle with a Main type	163
	Configuration of the Capture on a Main Type	164
20.2	Capture with an Encoder	165
	Capture with an Encoder	166
	Configuration of the Capture on an Encoder	169
Chapter 21	Preset and Enable Functions	171
	Preset Function	172
	Free-large or Period Meter Preset Conditions	174
	Enable Function	176
Appendices	177
Appendix A	General Information	179
	Dedicated Functions	180
	General Information on Administrative and Motion Function Block Management	181
Appendix B	Data Types	183
	IMMEDIATE_ERR_TYPE: Type for Error Variable of the GetImmediateValue Function Block	184
	EXPERT_ERR_TYPE: Type for Error Variable of EXPERT Function Block	185
	EXPERT_PARAMETER_TYPE: Type for Parameters to Get or to Set on EXPERTFunction Block	186
	EXPERT_REF: EXPERT Reference Value	187
	EXPERT_TIMEBASE_TYPE: Type for HSC Time Base Variable	188

Appendix C	Function Blocks	189
	EXPERTGetImmediateValue: Read Counter Value of HSC or Encoder Function	190
	EXPERTGetCapturedValue: Returns Content of Capture Registers. . . .	191
	EXPERTGetDiag: Provides Detail of Detected Error on a Principal EXPERT I/O Function	193
	EXPERTGetParam: Returns Parameters of Principal EXPERT I/O Function	196
	EXPERTSetParam: Adjust Parameters of a HSC	198
	Encoder_M258: Encoder Function Block	200
	HSCMain_M258: HSC Main Function Block	203
	HSCSimple_M258: HSC Simple Function Block	208
Appendix D	Function and Function Block Representation	211
	Differences Between a Function and a Function Block	212
	How to Use a Function or a Function Block in IL Language	213
	How to Use a Function or a Function Block in ST Language	216
Glossary	219
Index	249

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, 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 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 an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

WARNING

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

 **CAUTION**

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

NOTICE

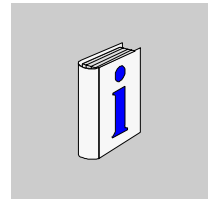
NOTICE is used to address practices not related to physical injury.

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 documentation will acquaint you with the High Speed Counter (HSC) functions and variables offered within the M258 controller.

This documentation describes the functions and variables of the M258 HSC library.

In order to use this manual, you must:

- Have a thorough understanding of the M258, including its 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 with the release of SoMachine V3.1.

Related Documents

Title of Documentation	Reference Number
Modicon M258 Logic Controller Programming Guide	EIO0000000402 (Eng), EIO0000000403 (Fre), EIO0000000404 (Ger), EIO0000000405 (Spa), EIO0000000406 (Ita), EIO0000000407 (Chs)
Modicon M258 Logic Controller Hardware Guide	EIO0000000432 (Eng), EIO0000000433 (Fre), EIO0000000434 (Ger), EIO0000000435 (Spa), EIO0000000436 (Ita), EIO0000000437 (Chs)

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

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

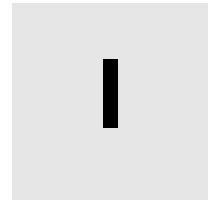
- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

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

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

High Speed Counter and Encoder Overview



Overview

This chapter provides an overview description, available modes, functionality and performances of the different HSC types and encoder types.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	Introduction	15
2	High Speed Counter Overview	25
3	Encoder Overview	31

Introduction



1

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Expert I/O Overview	16
Add an Expert function	19
Embedded Expert I/O mapping	22

Expert I/O Overview

Introduction

The controller base provides:

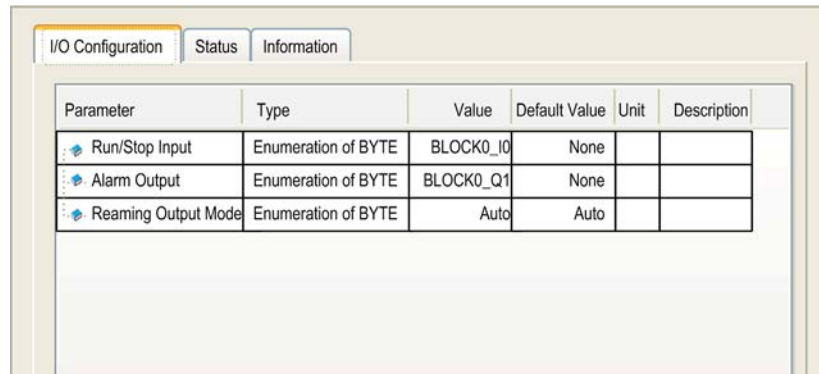
- 2 embedded expert I/O modules (DM72F0 and DM72F1) with:
 - 5 fast inputs
 - 2 regular inputs
 - 2 fast outputs
- 1 Controller Power Distribution Module (CPDM)

Each embedded expert I/O module (DM72F•) can support expert functions (see page 19).

Embedded Expert I/O Configuration

To configure the Expert I/O, proceed as follows:

Step	Action
1	Select the Configuration tab and double-click the controller.
2	Click the Expert I/O entry on the left hand side.



Parameter	Function
Run/Stop Input	Define one input to be used as Run/Stop input (see page 17).
Alarm Output	Define one output to be used as alarm output (see page 17).
Rearming Output Mode	Define the rearming output mode (see page 18).


Run/Stop Input

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

NOTE: Run/Stop input is managed even if the option **Update I/O while in stop** is not selected in Controller Device Editor (PLC setting tab) (see *Modicon M258 Logic Controller, Programming Guide*).

Inputs assigned to configured expert functions can not be configured as Run/Stop.

For further details about controller states and states transitions, refer to Controller State Diagram (see *Modicon M258 Logic Controller, Programming Guide*).

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

Alarm Output

This output is set logical 1 when the controller is in the RUNNING state and the application program is not stopped at a breakpoint.

Outputs assigned to configured expert functions can not be configured as the Alarm output.

NOTE:

The alarm output is set to 0 when:

- a task is stopped at a breakpoint, the alarm output signals that the controller has stopped executing the application.
- an error is detected on the expert IO (power failure, shortcut detection).

Rearming Output Mode

Fast outputs of DM72F• modules are push/pull technology. In case of detected error (short-circuit or over temperature), the output is put in tri-state and the condition is signaled by status bit (DM72F• channel IB1.0) and PLC_R.i_wLocalIOStatus (see *Modicon M258 Logic Controller, System Functions and Variables, Modicon M258 PLC System Library Guide*).

Two behaviors are possible:

- **Automatic rearming:** as soon as the detected error is corrected, the output is set again according to the current value assigned to it and the diagnostic value is reset.
- **Manual rearming:** when an error is detected, the status is memorized and the output is forced to tri-state until user manually clears the status (see I/O mapping channel).

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

WARNING

UNINTENDED MACHINE START-UP

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

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

Add an Expert function

Introduction

Each DM72F• expert module can support expert functions. Expert functions are defined as either simple or complex. Only one type can be configured per module:

- simple functions:
 - HSC Simple
 - Event_Latch I/O
- complex functions:
 - HSC Main
 - Encoder
 - PWM Generator
 - Frequency generator

When an I/O is not used by an expert function, it can be used as a regular I/O.

NOTE:

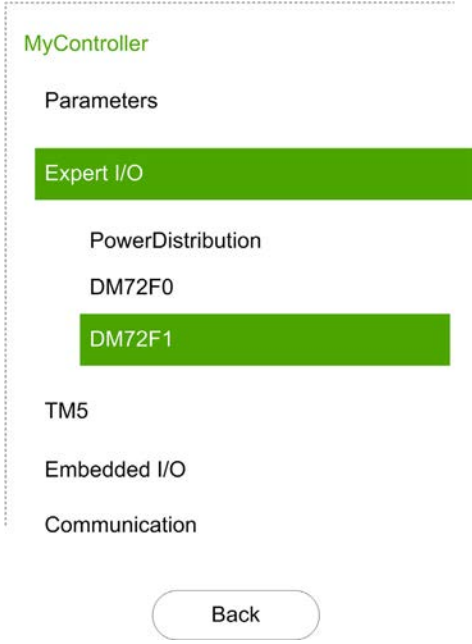
- When a regular input is used as Run/Stop, it can not be used by an expert function.
- When a regular output is used as Alarm, it can not be used by an expert function.

For more details, refer to Embedded expert I/O Configuration (*see page 16*).

Adding an Expert Function

To add an Expert function, proceed as follow:

Step	Action
1	Select the Configuration tab and double-click the controller.
2	Click the Expert I/O entry on the left hand side.

Step	Action
3	<p>Click the DM72F0 or DM72F1 sub-entry on the left hand side.</p> 
4	Click the Add Device button.
5	In the Add Device dialog box, select the expert function and click the Add and Close button.

The following expert functions can be added:

Function	Description	Refer to...
Event_Latch	With the Event_Latch function, the Embedded Expert inputs can be configured as event or latch.	Event_Latch configuration (see <i>Modicon M258 Logic Controller, Programming Guide</i>)
HSC	The HSC functions can execute fast counts of pulses from sensors, encoders, switches, etc. that are connected to dedicated fast inputs.	M258 HSC Library .

Function	Description	Refer to...
PWM Frequency Generator	The PWM function generates a square wave signal on dedicated output channels with a variable duty cycle. The Frequency Generator function generates a square wave signal on dedicated output channels with a fixed duty cycle (50%).	M258 PWM library (see <i>Modicon M258 Logic Controller, Pulse Width Modulation, M258 Expert I/O Library Guide</i>).
Encoder	The goal of this function is to connect an encoder to acquire a position. This function can be installed on an Embedded Expert I/O interface and supports only an incremental encoder. You can configure a linear or rotary axis.	M258 HSC Library

Expert Function Assignment

Expert functions assignment according to the interface (columns exclude each other):

I/F Interface	Expert Functions				
	Simple functions: ● Fast I/O: Event or latched ● HSC Simple	HSC_Main	Encoder	PWM	Frequency Generator
DM72F0	Up to 4	1	1	1	1
DM72F1	Up to 4	1	1	1	1

For more details, refer to Expert I/O Mapping (see page 22).

Expert Function I/O within Regular I/O

Expert Function I/O within Regular I/O

- Inputs can be read through memory variable standard even if configured in expert function
- An Input can not be configured in an expert function if it has already been configured as a Run/Stop.
- An Output can not be configured in an expert function if it has already been configured as a Alarm.
- %Q will not have any impact on reflex output.
- Short-Circuit management still applies on all outputs. Status of outputs are available.
- All I/O that are not used by expert functions are available as fast or regular I/O.

When inputs are used in expert functions (Latch, HSC,...), integrator filter is replaced by anti-bounce filter (see *Modicon M258, Logic Controller, Hardware Guide*). Filter value will be configured in expert function screen.

Embedded Expert I/O mapping

I/O mapping for Expert Function on DM72F•

Embedded Expert I/O mapping by expert function (M = Mandatory, C = depend on Configuration):

		I0	I1	I2	I3	I4	I5	Q0	Q1
Event_Latch 0/4	Input	M							
Event_Latch 1/5	Input		M						
Event_Latch 2/6	Input			M					
Event_Latch 3/7	Input				M				
HSC Simple 0/4	Input A	M							
HSC Simple 1/5	Input A		M						
HSC Simple 2/6	Input A			M					
HSC Simple 3/7	Input A				M				
HSC Main 0/1	Input A	M							
	Input B		C						
	SYNC			C					
	CAP				C				
	EN					C			
	REF						C		
	Outputs							C	C
PWM 0/1	Outputs							M	
	SYNC			C					
	EN					C			
Frequency Generator 0/1	Outputs							M	
	SYNC			C					
	EN					C			
Standard Encoder	Input A	M							
	Input B		M						
	SYNC			C					
	CAP				C				
	EN					C			
	REF						C		
	Outputs							C	C

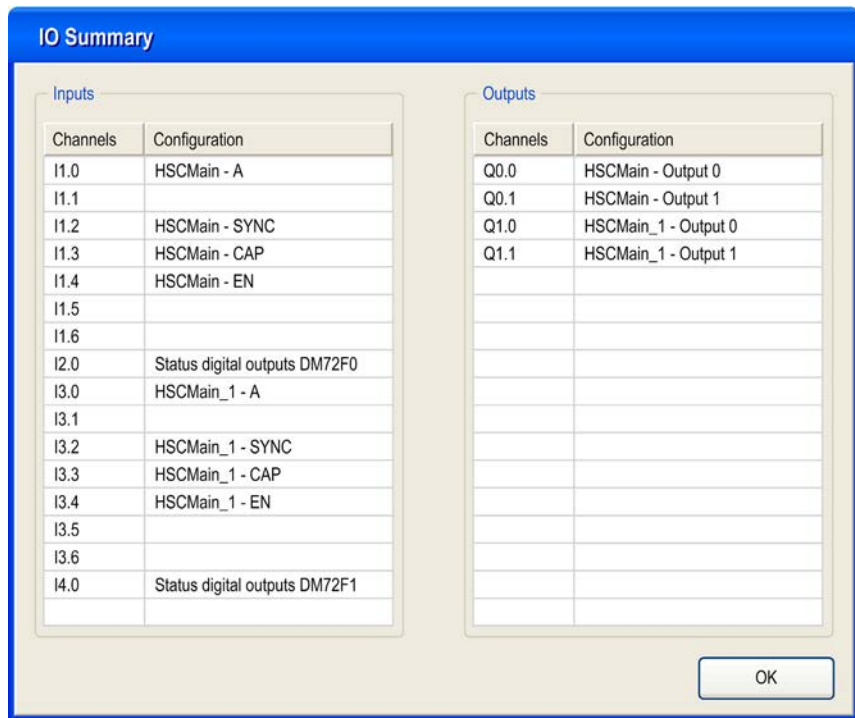
IO Summary

The IO Summary window displays the DM72F• IO mapping. You can see the I/O used by expert function.

The IO Summary window is accessible from Expert I/O or DM72F• entries:

Step	Action
1	Select the Configuration tab and double-click on your controller.
2	Click the Expert I/O entry in the left hand side. or Click the Expert I/O →DM72F• entry in the left hand side.
3	Click the Summary button.

Example of IO Summary:



The screenshot shows the 'IO Summary' window with two main sections: 'Inputs' and 'Outputs'. Each section contains a table with 'Channels' and 'Configuration' columns.

Inputs		Outputs	
Channels	Configuration	Channels	Configuration
I1.0	HSCMain - A	Q0.0	HSCMain - Output 0
I1.1		Q0.1	HSCMain - Output 1
I1.2	HSCMain - SYNC	Q1.0	HSCMain_1 - Output 0
I1.3	HSCMain - CAP	Q1.1	HSCMain_1 - Output 1
I1.4	HSCMain - EN		
I1.5			
I1.6			
I2.0	Status digital outputs DM72F0		
I3.0	HSCMain_1 - A		
I3.1			
I3.2	HSCMain_1 - SYNC		
I3.3	HSCMain_1 - CAP		
I3.4	HSCMain_1 - EN		
I3.5			
I3.6			
I4.0	Status digital outputs DM72F1		

An 'OK' button is located at the bottom right of the window.

High Speed Counter Overview

2

Overview

This chapter provides an overview of the different types of High Speed Counters (HSC).

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Simple Type Overview	26
Main Type Overview	27
Choosing your HSC	28

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, you cannot trigger an event or a reflex output.

Simple Type Modes

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

One-shot (*see page 37*): In this mode, the counter current value register decrements (from a user-defined value) for each pulse applied to A input, until the counter reaches 0.

Modulo-loop (*see page 59*): 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 an **Expert I/O** interface is 100 kHz.

Main Type Overview

Overview

The **Main6** fast inputs and 2 reflex outputs.

Main Type Modes

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

One-shot (see page 45): In this mode, the counter current value register decrements (from a user-defined value) for each pulse applied to A input, until the counter reaches a 0.

Modulo-loop (see page 67): In this mode, the counter repeatedly counts 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 85): In this mode, the counter behaves like a high range up and down counter. Can be used with one Encoder.

Event Counting (see page 99): In this mode, the counter accumulates a number of events that are received during a user-configured time base.

Frequency meter (see page 113): In this mode, the counter measures the frequency of events. Frequency is the number of events per second (Hz).

Period meter (see page 125): 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 current counting value
- up to 4 thresholds
- up to 4 events (1 by threshold) can be associated with external tasks
- up to 2 reflex outputs

Performance

The maximum frequency admissible on an **Expert I/O** interface is 100 kHz.

Choosing your HSC

HSC Matrix

The table below provides an overview of all the HSC available with their specifications according to the mode requested:

Mode	Feature	Simple Type	Main Type
One-shot	Counting mode	Count down	Count down
	Enable with an HSC physical input	No	Yes
	Synchronization / Preset with an HSC physical input	No	Yes
	Compare function	No	Yes, 4 thresholds, 2 outputs and events
	Capture function	No	Yes, 1 capture register
	Configuration tuning	-	Stop Event
Modulo-loop	Counting mode	Count down	Single phase Count up / down Pulse / direction Quadrature
	Enable with an HSC physical input	No	Yes
	Synchronization / Preset with an HSC physical input	No	Yes
	Compare function	No	Yes, 4 thresholds, 2 outputs and events
	Capture function	No	Yes, 1 capture register
	Configuration tuning	-	-
Free-large	Counting mode	-	Count up / down Pulse / direction Quadrature
	Enable with an HSC physical input	-	Yes
	Synchronization / Preset with an HSC physical input	-	Yes
	Compare function	-	Yes, 4 thresholds, 2 outputs and events
	Capture function	-	Yes, 1 capture register
	Configuration tuning	-	Limits management

Mode	Feature	Simple Type	Main Type
Event	Counting mode	-	Pulse counting during given period of time
	Enable with an HSC physical input	-	Yes
	Synchronization / Preset with an HSC physical input	-	Yes
	Compare function	-	No
	Capture function	-	No
	Configuration tuning	-	Time base
Frequency Meter	Counting mode	-	Pulse counting on time base
	Enable with an HSC physical input	-	Yes
	Synchronization / Preset with an HSC physical input	-	Yes
	Compare function	-	No
	Capture function	-	No
	Configuration tuning	-	-
Period Meter	Counting mode	-	Pulse counting on time base
	Enable with an HSC physical input	-	Yes
	Synchronization / Preset with an HSC physical input	-	Yes
	Compare function	-	No
	Capture function	-	No
	Configuration tuning	-	Resolution Time out

Encoder Overview

3

Standard Encoder Overview

Overview

The goal of this function is to connect an encoder to acquire a position.

This function can be installed on the embedded **Expert I/O** module (DM72F0 and DM72F1).

Standard Encoder Modes

When implemented on an **Expert I/O** interface, the encoder only supports the Incremental (*see Modicon LMC058 Motion Controller, High Speed Counting, LMC058 Expert I/O Library Guide*) mode.

On the **Encoder** interface, the power supply is monitored. A power supply error detected (*see page 193*) is cleared automatically when the power supply recovers.

Optional Feature

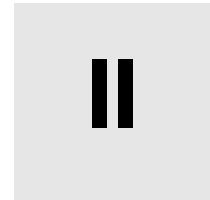
Optional features can be configured depending on the selected mode:

- hardware inputs to operate the counter (enable, preset) or capture the current counting value
- up to 4 thresholds
- up to 4 events (1 by threshold) can be associated with external tasks
- up to 2 reflex outputs

Performance

The maximum frequency admissible on an **Expert I/O** module is 100 kHz.

One-shot Mode



Overview

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

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
4	One-shot Mode Principle	35
5	One-shot With a Simple Type	37
6	One-shot With a Main Type	45

One-shot Mode Principle

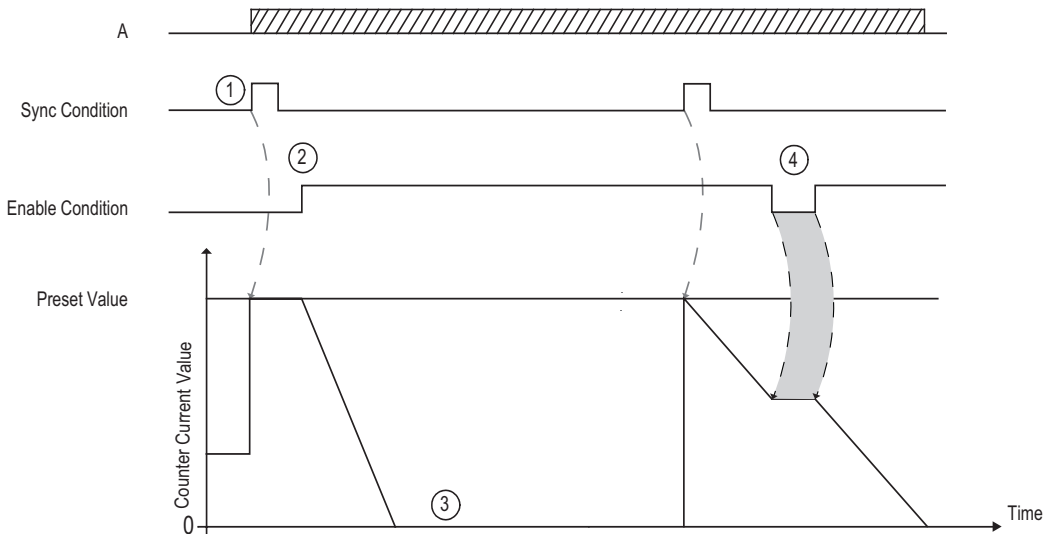


One-shot Mode Principle Description

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 current value. The counter stops when its current 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 current value) and the counter is activated.
2	When Enable condition = 1, the current 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 Enable condition = 0, the counter ignores the pulses from input A and retains its current 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 176) and Preset (see page 172) function.

One-shot With a Simple Type

5

Overview

This chapter describes how to implement a high speed counter in **One-shot** mode using a **Simple** type.

What's in this Chapter?

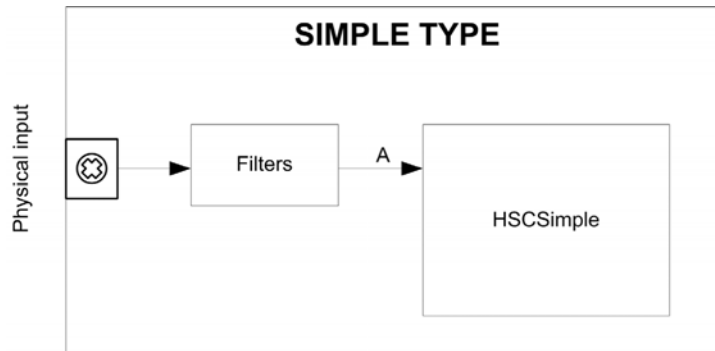
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	38
Configuration of the Simple Type in One-shot Mode	39
Programming the Simple Type	41
Adjusting Parameters	43

Synopsis Diagram

Synopsis Diagram

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

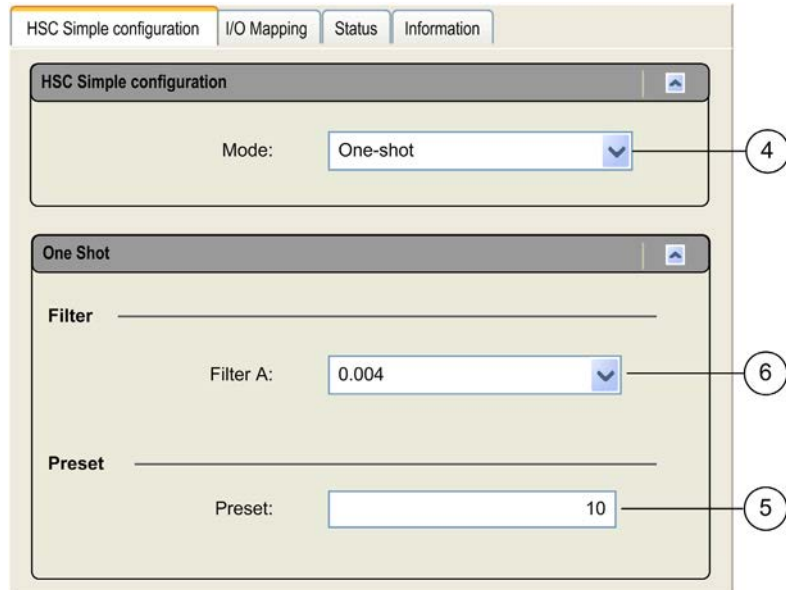


A is the counting input of the High Speed Counter.

Configuration of the Simple Type in One-shot Mode

Configuration Window

The figure below shows the **Simple** type in **One-shot** mode configuration window. The numbers in the bullets are associated with the Configuration Procedure table:



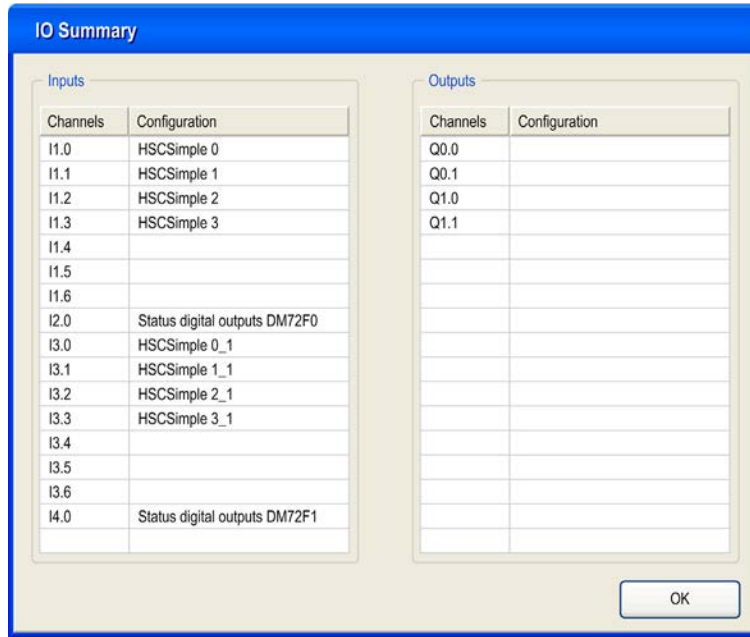
Configuration Procedure

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

Step	Action
1	Enter the Configuration screen.
2	Double click the controller.
3	Select Expert I/O → DM72F0/DM72F1 → HSCSimple Result: The HSC Simple configuration window opens.
4	Set the mode to One-shot from the drop down menu, by selecting Mode → One-shot
5	Set the preset value from One-shot → Preset → Preset
6	Set the anti-bounce filter value from the drop down menu, by selecting One-shot → Filter → Filter A

IO Summary

The input/output configuration is displayed in the IO Summary window, opened with the **Summary** button:



Refer to the hardware guide for wiring details. (see *Modicon M258, Logic Controller, Hardware Guide*)

Programmable Filter

The filtering value on the **Simple** type input determines the counter maximum frequency as shown in the table below:

Input	Filter value	Maximum counter frequency
A	0.002 ms	200 kHz
	0.004 ms	100 kHz
	0.012 ms	40 kHz
	0.04 ms	10 kHz
	0.12 ms	4 kHz
	0.4 ms	1 kHz
	1.2 ms	400 Hz
	4 ms	100 Hz


Programming the Simple Type

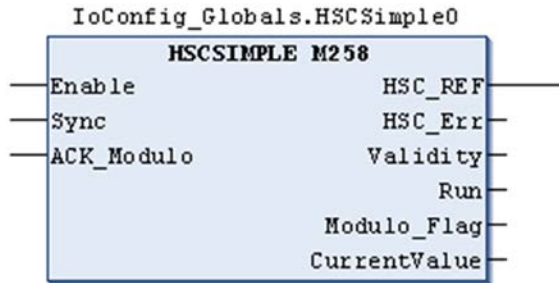
Overview

A **Simple** type is always managed by an HSCSimple (see page 208) function block.

NOTE: At build, an error code is given if the HSCSimple function block is used to manage a different HSC type.

Adding a HSCSimple Function Block

Step	Description
1	Insert the HSCSimple_M258 function block with the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP →HSC →HSCSimple_M258
2	Type the Simple type instance name (defined in configuration, step 5) or look for the function block instance by clicking on:  Using the input assistant, the HSC instance can be selected at the following path: Global Variables →<MyController> →PLC Logic → IoConfig_Globals



I/O Variables Usage

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

The following table describes the input variables:

Input	Type	Comment
Enable	BOOL	TRUE = authorizes changes to the current counter value.
Sync	BOOL	On rising edge, presets and starts the counter
ACK_Modulo	BOOL	Not used

The following table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF (see page 187)	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the Administrative function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (see page 193) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that the output values on the function block are valid.
Run	BOOL	TRUE = counter is running. Switches to 0 when CurrentValue reaches 0. A rising edge on Sync is needed to restart the counter.
Modulo_Flag	BOOL	Not relevant
CurrentValue	DWORD	Current count value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table below can be read or modified by using the `EXPERTGetParam` (see page 196) or `EXPERTSetParam` (see page 198) function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M258 Logic Controller, Programming Guide*).

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 186) that can be read or modified while the program is running:

Parameter	Description
<code>EXPERT_PRESET</code>	to get or set the Preset value of an HSC

One-shot With a Main Type



6

Overview

This chapter describes how to implement a high speed counter in **One-shot** mode using a **Main** type.

What's in this Chapter?

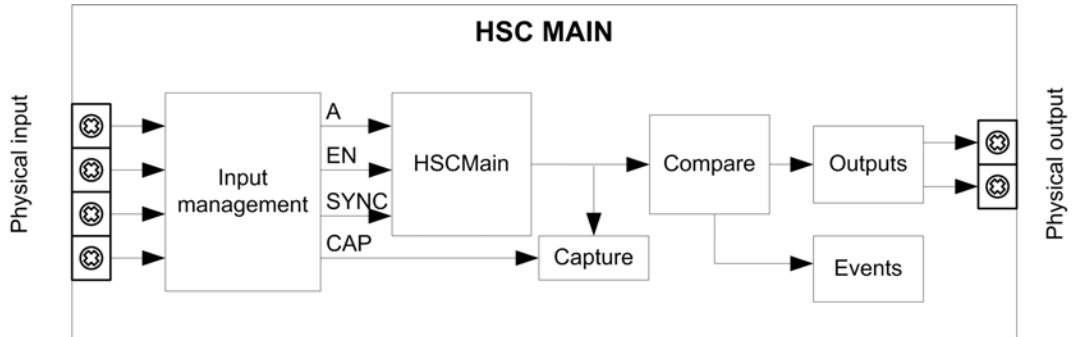
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	46
Configuration of the Main Type in One-shot Mode	47
Programming the Main Type	50
Adjusting Parameters	53

Synopsis Diagram

Synopsis Diagram

The following 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.

CAP is the capture input of the counter.

SYNC is the synchronization input of the counter.

Optional Function

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

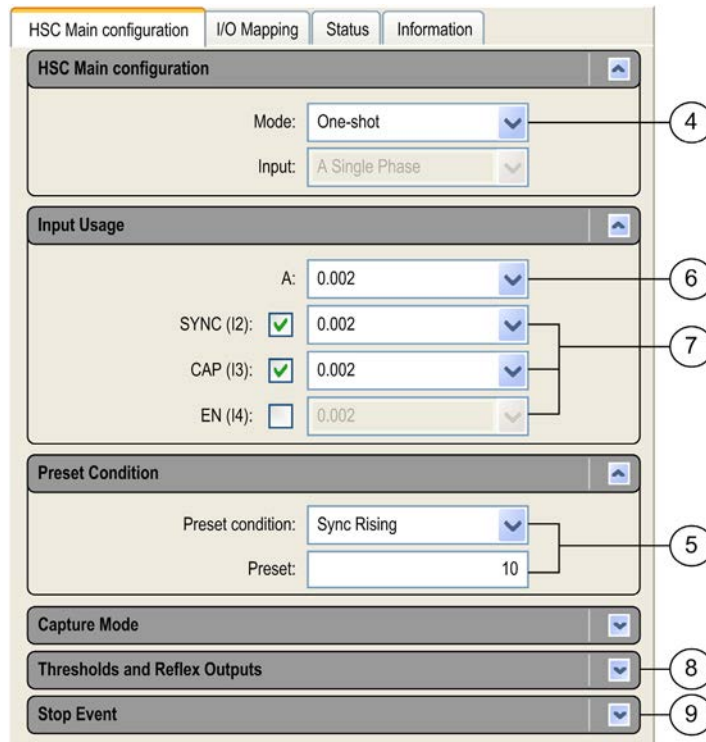
- Compare (see page 153)
- Capture (see page 162)
- Enable by a physical input (see page 176)
- Synchronize by a physical input (see page 172)

Configuration of the Main Type in One-shot Mode

Configuration Window

The figure below shows the **Main** type in **One-shot** mode configuration window.

The numbers in the bullets are associated with the Configuration Procedure table:



Configuration Procedure

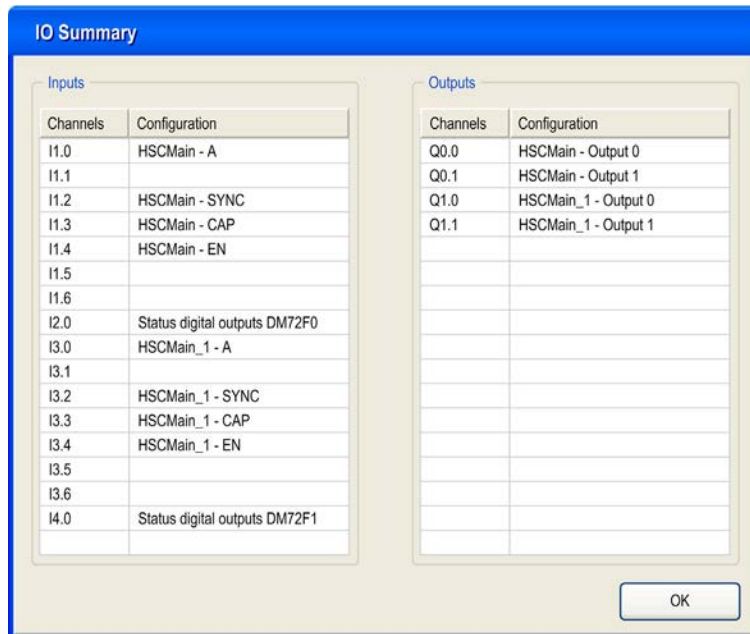
Follow this procedure to configure a **Main** type in **One-shot** mode:

Step	Action
1	Enter the Configuration Window
2	Double click the controller
3	Select Expert I/O → DM72F0/DM72F1 → HSCMain Result: The HSC Main configuration window opens.
4	Set the mode to One-shot from the drop down menu, by selecting HSC Main configuration → Mode → One-shot

Step	Action
5	Set the preset condition from the drop down menu Preset Condition → Preset condition → One-shot and provide a value to the Preset field.
6	Set the anti-bounce filtering value from the drop down menu, by selecting Input Usage → A
7	Optionally, select the SYNC (I2) , CAP (I3) and EN (I4) check boxes and provides a value from the associated drop down menu to enable the Synchronization function (see page 172), Enable function (see page 176) and Capture function (see page 162) with a physical input.
8	Optionally, select the TH0 check box to provide a threshold value. This enables the Compare function and reflex outputs can be configured (see page 153)
9	When Stop Event is set to Yes, the external event (BLOCK0_HSCSTOP or BLOCK1_HSCSTOP) must be used to trigger an external task (see page 159).

IO Summary

The input/output configuration is displayed in the IOsummary window, opened with the **Summary** button:



Refer to the hardware guide for wiring details. (see *Modicon M258, Logic Controller, Hardware Guide*)

Programmable Filter

The filtering value on the **Main** type input determines the counter maximum frequency as shown in the table below:

Input	Filter value	Maximum counter frequency
A	0.002 ms	200 kHz
	0.004 ms	100 kHz
	0.012 ms	40 kHz
	0.04 ms	10 kHz
	0.12 ms	4 kHz
	0.4 ms	1 kHz
	1.2 ms	400 Hz
	4 ms	100 Hz


Programming the Main Type

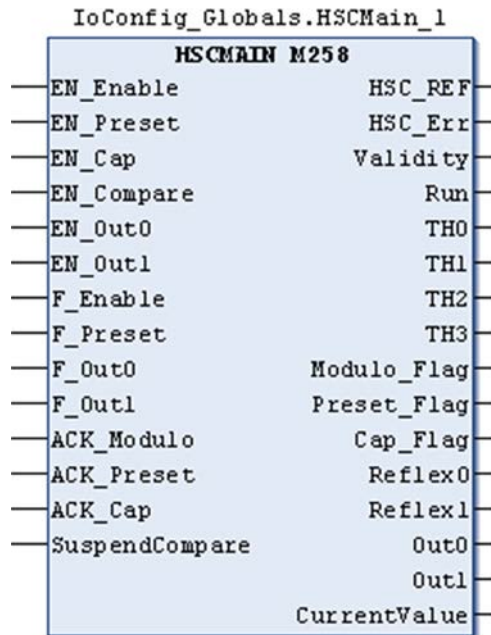
Overview

Main type is always managed by an HSCMain function block.

NOTE: At build, an error code is given if the HSCMain function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Insert the HSCMain_M258 function block with the insert box assistant. The function block can be found at this path: Function Block (Libraries) →SEC_EXP →HSC →HSCMain_M258
2	Type the Main type instance name (defined in configuration, step 5) or look for the function block instance by clicking on:  Using the input assistant, the HSC instance can be selected at the following path: Global Variables →<MyController> →PLC Logic →IoConfig_Globals



I/O Variables Usage

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

The following table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When EN input is configured: if TRUE, authorizes the counter enable via the Enable input (<i>see page 176</i>).
EN_Preset	BOOL	When SYNC input is configured: if TRUE, authorizes the counter synchronization and start via the Sync input (<i>see page 172</i>).
EN_Cap	BOOL	When CAP input is configured: if TRUE, enables the Capture input (<i>see page 162</i>).
EN_Compare	BOOL	TRUE = enables the comparator operation (<i>see page 153</i>) (using Thresholds 0, 1, 2, 3): <ul style="list-style-type: none"> ● basic comparison (TH0, TH1, TH2, TH3 output bits) ● reflex (Reflex0, Reflex1 output bits) ● events (to trigger external tasks on threshold crossing)
EN_Out0	BOOL	TRUE = enables physical output Output0 to echo the Reflex0 value (if configured).
EN_Out1	BOOL	TRUE = enables physical output Output1 to echo the Reflex1 value (if configured).
F_Enable	BOOL	Forces the Enable condition (<i>see page 176</i>).
F_Preset	BOOL	Forces the Preset condition (<i>see page 172</i>).
F_Out0	BOOL	TRUE = forces Output0 to 1 (if Reflex0 is configured).
F_Out1	BOOL	TRUE = forces Output1 to 1 (if Reflex1 is configured).
ACK_Modulo	BOOL	Not used
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: <ul style="list-style-type: none"> ● TH0, TH1, TH2, TH3, Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value. ● Physical Outputs Output0 and Output1 maintain their last value. ● Events are masked. <p>NOTE: EN_Compare, EN_Out0, EN_Out1, F_Out0, F_Out1 remain operational while SuspendCompare is set.</p>

The following table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF (see page 187)	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the Administrative function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (see page 193) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	TRUE = counter is running. Switches to 0 when CurrentValue reaches 0. A rising edge on Sync is needed to restart the counter.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 (see page 153).
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 (see page 153).
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (see page 153).
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (see page 153).
Modulo_Flag	BOOL	Not relevant
Preset_Flag	BOOL	Set to 1 by the preset of the counter (see page 172).
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register (see page 162). This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0. (see page 153)
Reflex1	BOOL	State of Reflex1. (see page 153)
Out0	BOOL	State of physical output Output0 (if Reflex0 configured).
Out1	BOOL	State of physical output Output1 (if Reflex1 configured).
CurrentValue	DINT	Current counter value of the counter.

Adjusting Parameters

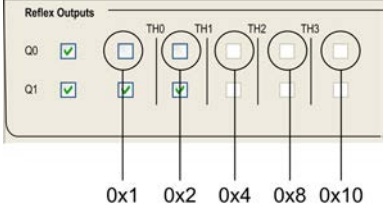
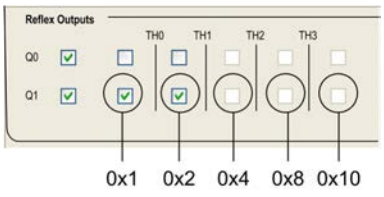
Overview

The list of parameters described in the table below can be read or modified by using the `EXPERTGetParam` (see page 196) or `EXPERTSetParam` (see page 198) function blocks.

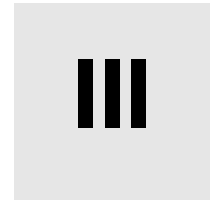
NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M258 Logic Controller, Programming Guide*).

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 186) which can be read or modified while the program is running:

Parameter	Description
<code>EXPERT_PRESET</code>	to get or set the Preset value of an HSC
<code>EXPERT_THRESHOLD0</code>	to get or set the Threshold 0 value of an HSC
<code>EXPERT_THRESHOLD1</code>	to get or set the Threshold 1 value of an HSC
<code>EXPERT_THRESHOLD2</code>	to get or set the Threshold 2 value of an HSC
<code>EXPERT_THRESHOLD3</code>	to get or set the Threshold 3 value of an HSC
<code>EXPERT_REFLEX0</code>	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: 
<code>EXPERT_REFLEX1</code>	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: 

Modulo-loop Mode



Overview

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

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
7	Modulo-loop Principle	57
8	Modulo-loop with a Simple Type	59
9	Modulo-loop With a Main Type	67

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 172*), the counter is activated and the current value is reset to 0.

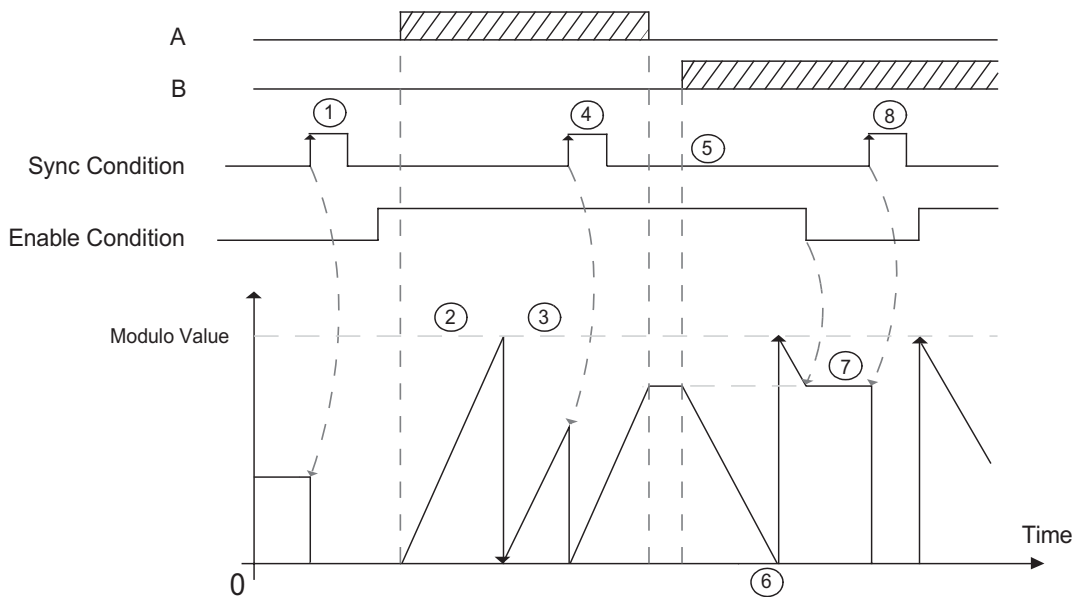
When counting is enabled (*see page 176*):

Incrementing direction: the counter increments until it reaches the modulo value.

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.

Principle Diagram



Stage	Action
1	On the rising edge of Sync condition, the current 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 current 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 current counter value is reset to 0.

NOTE: Enable and Sync conditions depends on configuration. These are described in the Enable (see page 176) and Preset (see page 172) function.

Modulo-loop with a Simple Type



8

Overview

This chapter describes how to implement a high speed counter in **Modulo-loop** mode using a **Simple** type.

What's in this Chapter?

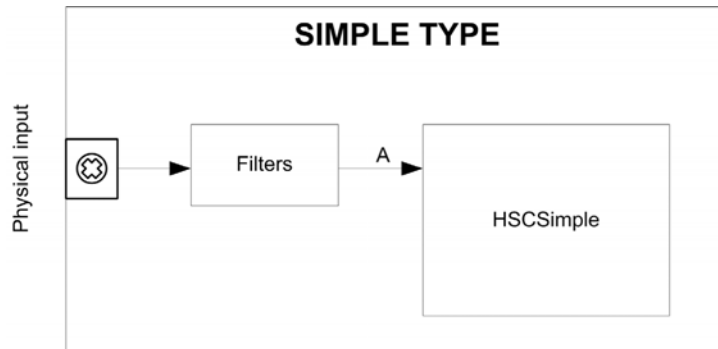
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	60
Configuration of the Simple Type in Modulo-loop Mode	61
Programming the Simple Type	63
Adjusting Parameters	65

Synopsis Diagram

Synopsis Diagram

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



A is the counting input of the High Speed Counter.

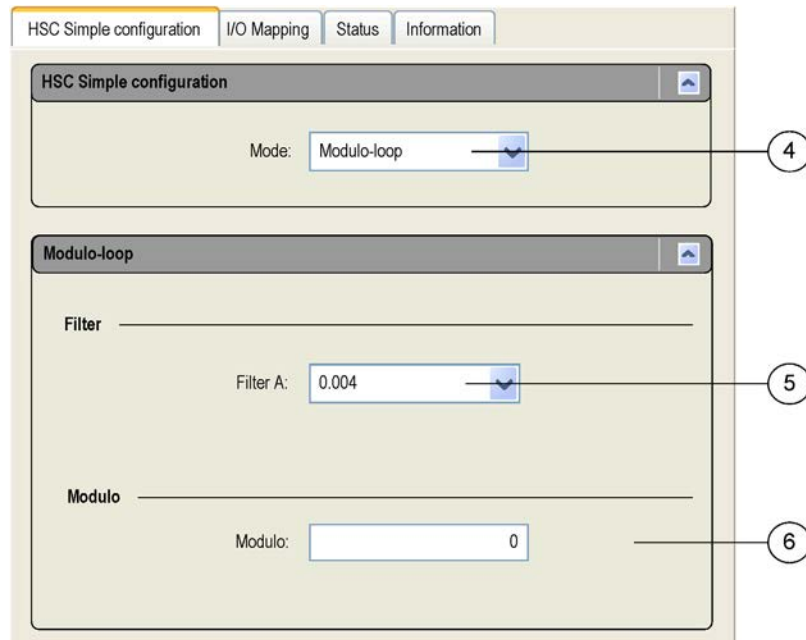
A **Simple** type can only count up.

Configuration of the Simple Type in Modulo-loop Mode

Configuration Window

The following figure shows the **Simple** type in **Modulo-loop** mode configuration window.

The numbers in the bullets are associated with the Configuration Procedure table:



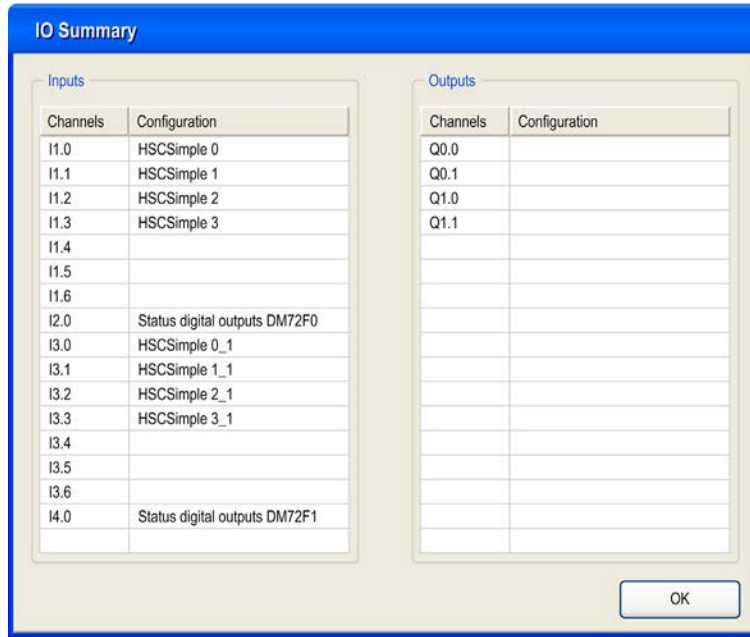
Configuration Procedure

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

Step	Action
1	Enter the Configuration Window .
2	Double click the controller.
3	Select Expert I/O → DM72F0/DM72F1 → HSCSimple Result: The HSC Simple configuration window opens.
4	Set the mode to Modulo-loop from the drop down menu, by selecting HSC Simple configuration → Mode → Modulo-loop
5	Set the anti-bounce filtering value from the drop down menu, by selecting Modulo-loop → Filter → Filter A
6	Set the modulo value from Modulo-loop → Modulo

IO Summary

The input/output configuration is displayed in the IO Summary window, opened with the **Summary** button:



Refer to the hardware guide for wiring details. (see *Modicon M258, Logic Controller, Hardware Guide*)

Programmable Filter

The filtering value on the **Simple** type input determines the counter maximum frequency as shown in the table below:

Input	Filter value	Maximum counter frequency
A	0.002 ms	200 kHz
	0.004 ms	100 kHz
	0.012 ms	40 kHz
	0.04 ms	10 kHz
	0.12 ms	4 kHz
	0.4 ms	1 kHz
	1.2 ms	400 Hz
	4 ms	100 Hz

Programming the Simple Type

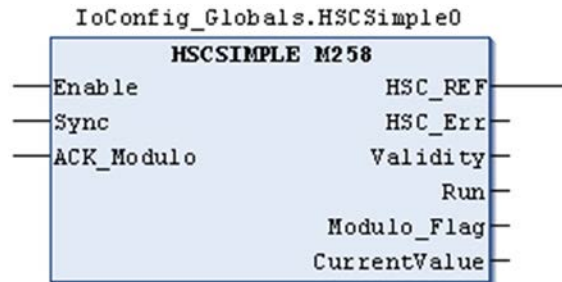
Overview

A **Simple** type is always managed by an HSCSimple (see page 208) function block.

NOTE: At build, an error code is given if the HSCSimple function block is used to manage a different HSC type.

Adding a HSCSimple Function Block

Step	Description
1	Insert the HSCSimple_M258 function block with the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP →HSC →HSCSimple_M258
2	Type the Simple type instance name (defined in configuration, step 5) or look for the function block instance by clicking on: <div style="border: 1px solid gray; padding: 2px; width: fit-content; margin: 5px 0;"> ??? ... </div> Using the input assistant, the HSC instance can be selected at the following path: Global Variables →<MyController> →PLC Logic → IoConfig_Globals



I/O Variables Usage

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

The following table describes the input variables:

Input	Type	Comment
Enable	BOOL	TRUE = authorizes changes to the current counter value.
Sync	BOOL	On rising edge, presets and starts the counter.
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag.

The following table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF (see page 187)	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the Administrative function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. EXPERTGetDiag (see page 193) function block may be used to get more information about this detected error.
Validity	BOOL	TRUE = indicates that the output values on the function block are valid.
Run	BOOL	Not relevant
Modulo_Flag	BOOL	Set to 1 when the counter roll overs the modulo.
CurrentValue	DWORD	Current count value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table below can be read or modified by using the `EXPERTGetParam` (see page 196) or `EXPERTSetParam` (see page 198) function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M258 Logic Controller, Programming Guide*).

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 186) that can be read or modified while the program is running:

Parameter	Description
<code>EXPERT_MODULO</code>	to get or set the modulo value of an HSC

Modulo-loop With a Main Type

9

Overview

This chapter describes how to implement a high speed counter in **Modulo-loop** mode using a **Main** type.

What's in this Chapter?

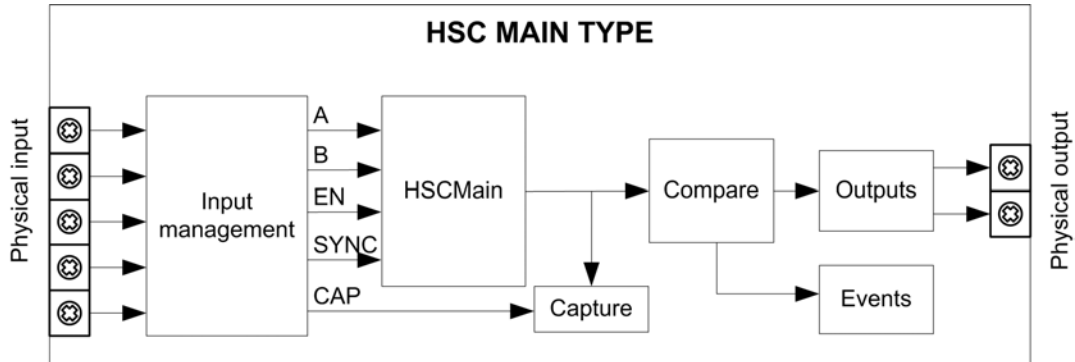
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	68
Configuration of the Main Type in Modulo-loop Mode	69
Programming the Main Type	72
Adjusting Parameters	75

Synopsis Diagram

Synopsis Diagram

The following 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.

CAP is the capture input of the counter.

SYNC is the synchronization input of the counter.

Optional Function

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

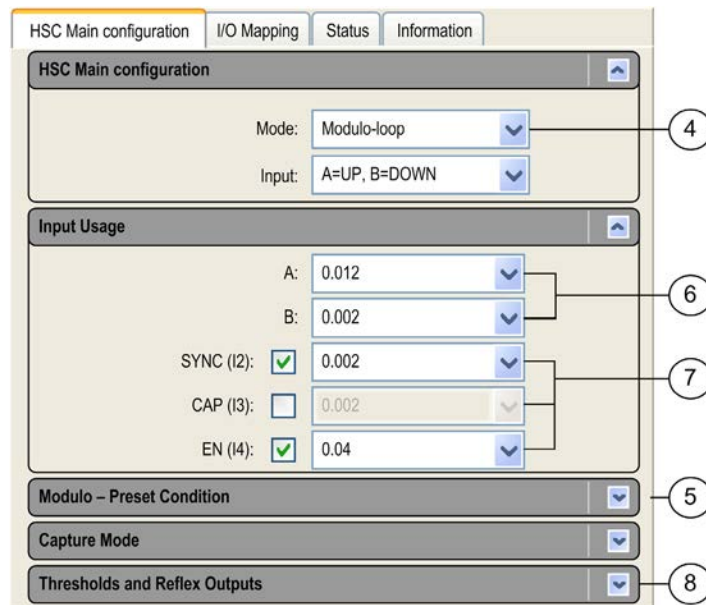
- Compare (see page 153)
- Capture (see page 162)
- Enable by a physical input (see page 176)
- Synchronize by a physical input (see page 172)

Configuration of the Main Type in Modulo-loop Mode

Configuration Window

The following figure shows the **Main** type in **Modulo-loop** mode configuration window.

The numbers in the bullets are associated with the Configuration Procedure table:



Configuration Procedure

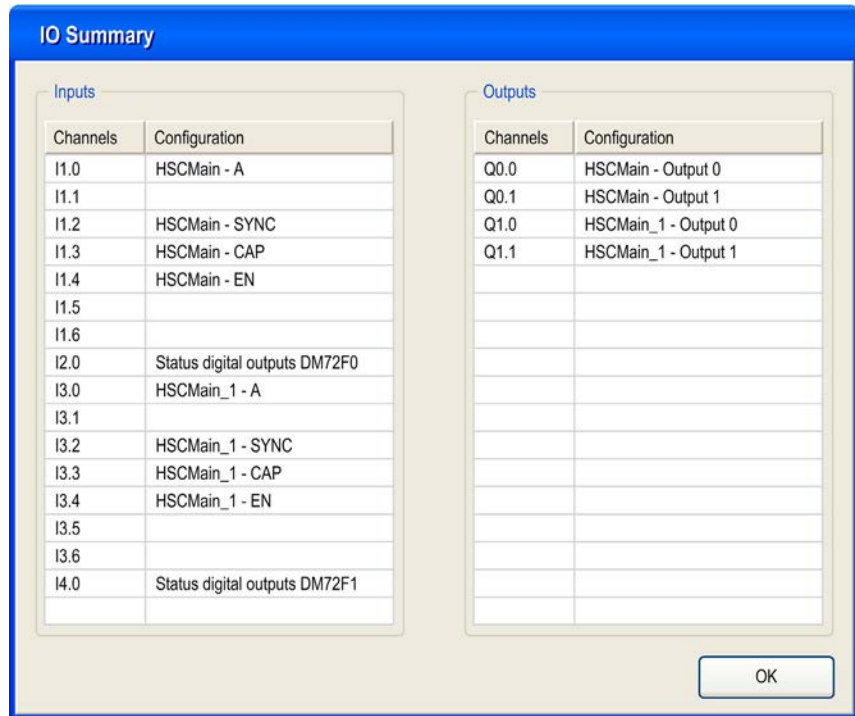
Follow this procedure to configure a **Main** type:

Step	Action
1	Enter the Configuration Window
2	Double click the controller
3	Select Expert I/O → DM72F0/DM72F1 → HSCMain Result: The HSC Main configuration window opens.
4	Set the mode to Modulo-loop from the drop down menu, by selecting HSC Main configuration → Mode → Modulo-loop Define your Input , this determines the action of the second input B.
5	Set the preset condition from the drop down menu, selected in Modulo - Preset Condition → Preset condition and provide a value to the Modulo field.

Step	Action
6	Set the anti-bounce filtering value from the drop down menu, by selecting Input Usage → A (and B Filter when applicable).
7	Optionally, select the SYNC (I2) , CAP (I3) and EN (I4) check boxes and provides a value from the associated drop down menu to enable the Synchronization function (see page 172), Enable function (see page 176) and Capture function (see page 162) with a physical input.
8	Optionally, select the TH0 check box to provide a threshold value. This enables the Compare function and reflex outputs can be configured (see page 153).

IO Summary

The input/output configuration is displayed in the IOsummary window, opened with the **Summary** button:



Refer to the hardware guide for wiring details. (see *Modicon M258, Logic Controller, Hardware Guide*)

Programmable Filter

The filtering value on the **Main** type input determines the counter maximum frequency as shown in the table below:

Input	Filter value	Maximum counter frequency
A, B	0.002 ms	200 kHz
	0.004 ms	100 kHz
	0.012 ms	40 kHz
	0.04 ms	10 kHz
	0.12 ms	4 kHz
	0.4 ms	1 kHz
	1.2 ms	400 Hz
	4 ms	100 Hz


Programming the Main Type

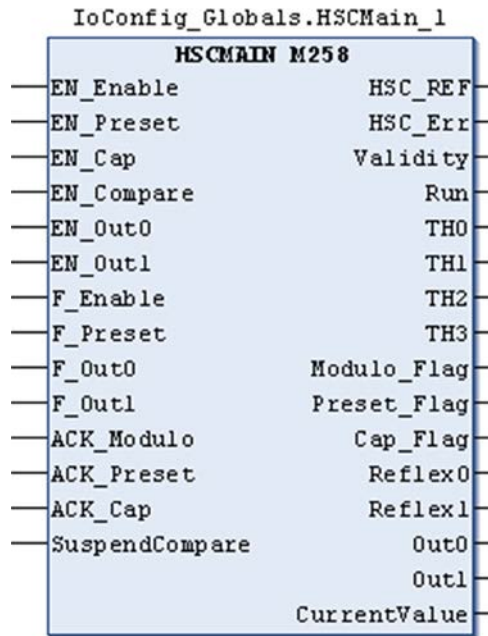
Overview

Main type is always managed by an HSCMain function block.

NOTE: At build, an error code is given if the HSCMain function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Insert the HSCMain_M258 function block with the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP → HSC → HSCMain_M258
2	Type the Main type instance name (defined in configuration, step 5) or look for the function block instance by clicking on:  Using the input assistant, the HSC instance can be selected at the following path: Global Variables → <MyController> → PLC Logic → IoConfig_Globals



I/O Variables Usage

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

The following table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When EN input is configured: if TRUE, authorizes the counter enable via the Enable input (<i>see page 176</i>).
EN_Preset	BOOL	When SYNC input is configured: if TRUE, authorizes the counter synchronization and start via the Sync input (<i>see page 172</i>).
EN_Cap	BOOL	When CAP input is configured: if TRUE, enables the Capture input (<i>see page 162</i>).
EN_Compare	BOOL	TRUE = enables the comparator operation (<i>see page 153</i>) using Thresholds 0, 1, 2, 3: <ul style="list-style-type: none"> ● basic comparison (TH0, TH1, TH2, TH3 output bits) ● reflex (Reflex0, Reflex1 output bits) ● events (to trigger external tasks on threshold crossing)
EN_Out0	BOOL	TRUE = enables physical output Output0 to echo the Reflex0 value (if configured).
EN_Out1	BOOL	TRUE = enables physical output Output1 to echo the Reflex1 value (if configured).
F_Enable	BOOL	Forces the Enable condition (<i>see page 176</i>).
F_Preset	BOOL	Forces the Preset condition (<i>see page 172</i>).
F_Out0	BOOL	TRUE = forces Output0 to 1 (if Reflex0 is configured).
F_Out1	BOOL	TRUE = forces Output1 to 1 (if Reflex1 is configured).
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: <ul style="list-style-type: none"> ● TH0, TH1, TH2, TH3, Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value. ● Physical outputs Output0 and Output1 maintain their last value ● Events are masked <p>NOTE: EN_Compare, EN_Out0, EN_Out1, F_Out0, F_Out1 remain operational while SuspendCompare is set.</p>

The following table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF (see page 187)	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the Administrative function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (see page 193) function block used to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	TRUE = counter is running.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 (see page 153).
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 (see page 153).
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (see page 153).
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (see page 153).
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 172).
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register (see page 162). This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0. (see page 153)
Reflex1	BOOL	State of Reflex1. (see page 153)
Out0	BOOL	State of physical output Output0 (if Reflex0 configured).
Out1	BOOL	State of physical output Output1 (if Reflex1 configured).
CurrentValue	DINT	Current counter value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table below can be read or modified by using the `EXPERTGetParam` (see page 196) or `EXPERTSetParam` (see page 198) function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M258 Logic Controller, Programming Guide*).

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 186) that can be read or modified while the program is running:

Parameter	Description
<code>EXPERT_MODULO</code>	to get or set the Modulo value of an HSC
<code>EXPERT_THRESHOLD0</code>	to get or set the Threshold 0 value of an HSC
<code>EXPERT_THRESHOLD1</code>	to get or set the Threshold 1 value of an HSC
<code>EXPERT_THRESHOLD2</code>	to get or set the Threshold 2 value of an HSC
<code>EXPERT_THRESHOLD3</code>	to get or set the Threshold 3 value of an HSC
<code>EXPERT_REFLEX0</code>	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: <div data-bbox="622 906 1002 1117" data-label="Diagram"> <p>The diagram shows a control panel for 'Reflex Outputs'. It has two rows of checkboxes: Q0 and Q1. Each row has five columns corresponding to thresholds TH0, TH1, TH2, TH3 and bit weights 0x1, 0x2, 0x4, 0x8, 0x10. In the Q0 row, the checkboxes for TH0, TH1, TH2, and TH3 are checked. In the Q1 row, the checkboxes for 0x1 and 0x2 are checked.</p> </div>
<code>EXPERT_REFLEX1</code>	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: <div data-bbox="622 1203 1002 1386" data-label="Diagram"> <p>The diagram shows a control panel for 'Reflex Outputs'. It has two rows of checkboxes: Q0 and Q1. Each row has five columns corresponding to thresholds TH0, TH1, TH2, TH3 and bit weights 0x1, 0x2, 0x4, 0x8, 0x10. In the Q0 row, the checkboxes for TH0, TH1, TH2, and TH3 are checked. In the Q1 row, the checkboxes for 0x1 and 0x2 are checked.</p> </div>

Free-large Mode



Overview

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

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
10	Free-large Mode Principle	79
11	Free-large With a Main Type	85

Free-large Mode Principle

10

Overview

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

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Free-large Mode Principle Description	80
Limits Management	83

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 176), the counter counts as follows in:

Incrementing direction: the counter increments.

Decrementing direction: the counter decrements.

The counter is activated by a preset edge (see page 174) which loads the preset value.

The current counter is stored in the capture register by using the Capture (see page 161) function.

If the counter reaches the counting limits, the counter will react according to the Limits Management (see page 83) configuration.

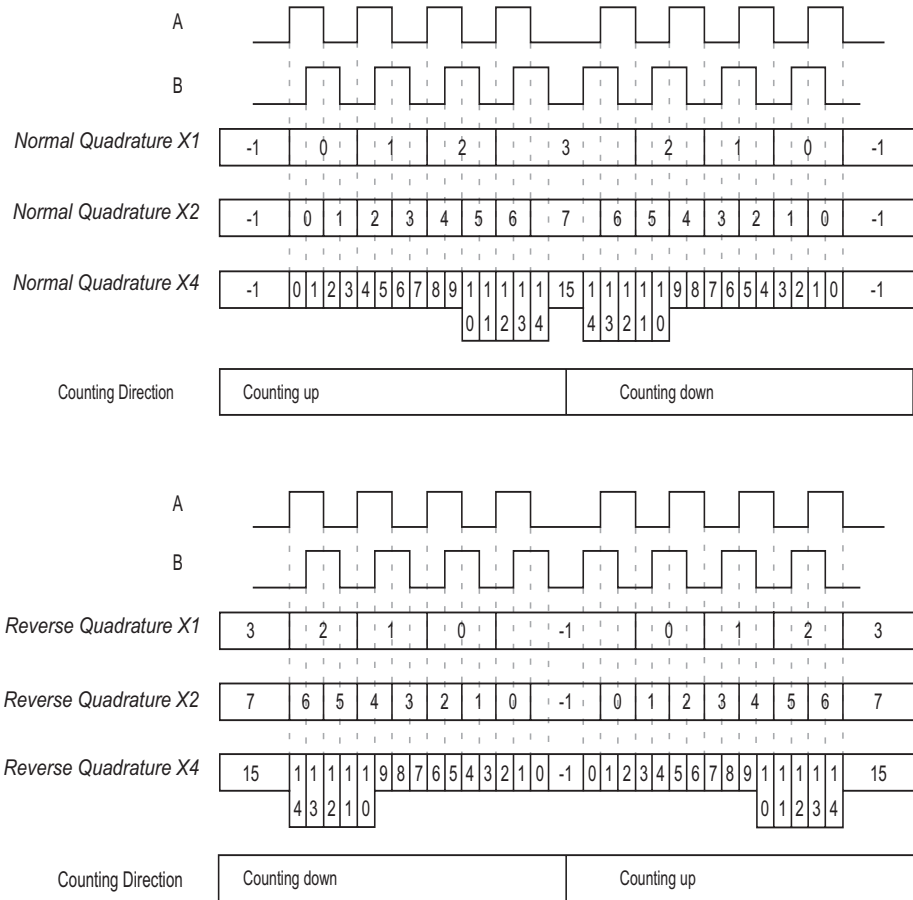
Input Modes

The following 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: <ul style="list-style-type: none"> ● 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	

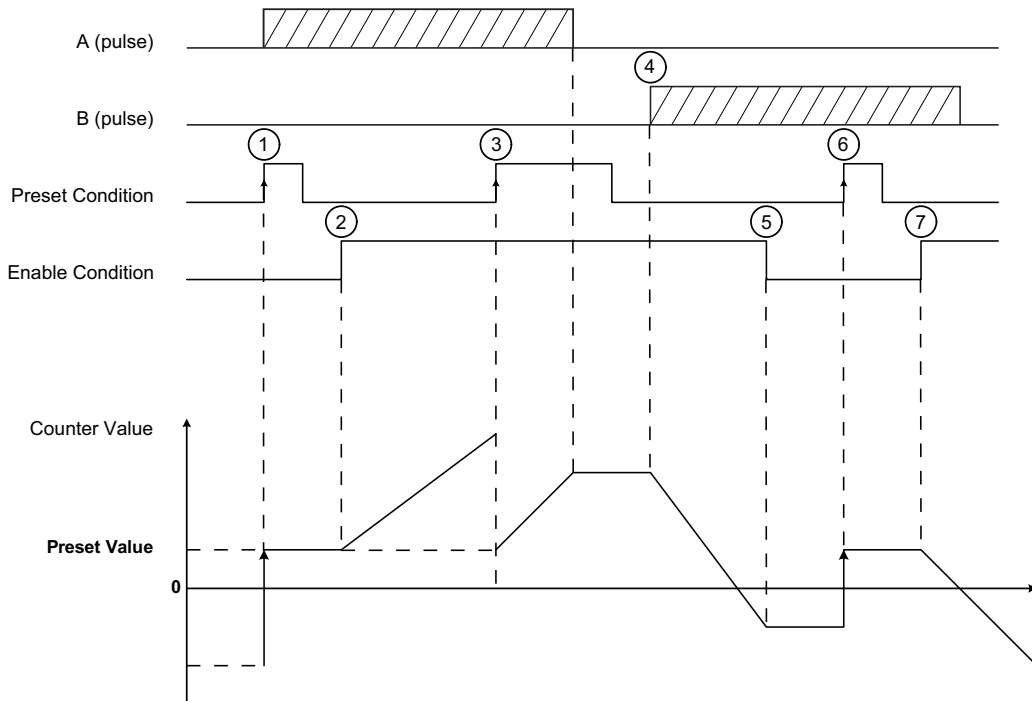
Quadrature Principle Diagram

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



Principle Diagram

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



Stage	Action
1	On the rising edge of Preset condition, the current 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 current 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 current value is set to the preset value.
7	When Enable condition = 1, the pulses on B decrements the counter value.

Limits Management

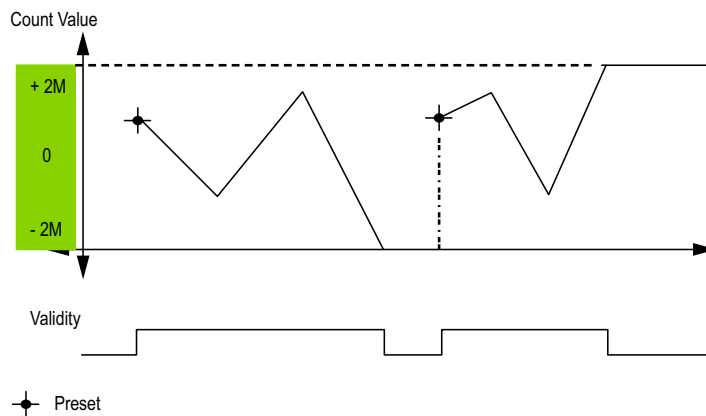
Overview

When the counter limit is reached, the counter can have 2 behaviors depending on configuration:

- Lock on limits
- Rollover

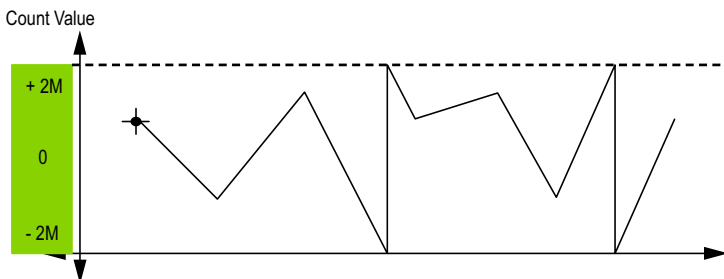
Lock on Limits

In the case of overflow or underflow counter, the current counter value is maintained to the limit value, the validity bit goes to 0, and the `Error` bit indicates this detected error until the counter is preset again.



Rollover

In the case of overflow or underflow of the counter, the current counter value goes automatically to the opposite limit value. `Modulo_Flag` (resp. `Overflow_Flag` for Encoder) output is set to 1.



Free-large With a Main Type

11

Overview

This chapter describes how to implement a high speed counter in **Free-large** mode using a **Main** type.

What's in this Chapter?

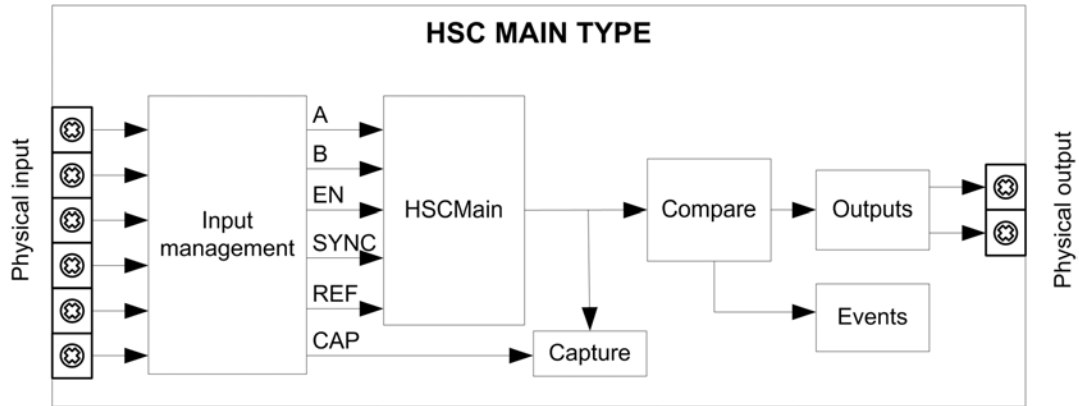
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	86
Configuration of the Main Type in Free-large Mode	87
Programming the Main Type	90
Adjusting Parameters	93

Synopsis Diagram

Synopsis Diagram

The following 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.

REF is the reference input of the counter.

CAP is the capture input of the counter.

SYNC is the synchronization input of the counter.

Optional Function

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

- Compare (*see page 153*)
- Capture (*see page 162*)
- Enable by a physical input (*see page 176*)
- Synchronize by a physical input (*see page 174*)

Configuration of the Main Type in Free-large Mode

Configuration Window

The following figure shows the **Main** type in **Free-large** mode configuration window.

The numbers in the bullets are associated with the Configuration Procedure table:

The screenshot displays the configuration window for the HSC Main type in Free-large mode. The window is organized into several sections, each with a title bar and a set of configuration options:

- HSC Main configuration:** Contains a 'Mode' dropdown set to 'Free-large' (callout 4) and an 'Input' dropdown set to 'A=UP, B=DOWN'.
- Input Usage:** Contains several input parameters, each with a checkbox and a value dropdown:
 - A: 0.002 (callout 6)
 - B: 0.002
 - SYNC (I2): 0.002
 - CAP (I3): 0.002
 - EN (I4): 0.002
 - REF (I5): 0.002
 A bracket groups the SYNC, CAP, EN, and REF parameters, with callout 7 pointing to this group.
- Preset Condition:** Contains a 'Preset condition' dropdown set to 'REF Rising' (callout 5) and a 'Preset' input field with the value '0'.
- Capture Mode:** A dropdown menu.
- Thresholds and Reflex Outputs:** A dropdown menu (callout 8).
- Limits:** A dropdown menu (callout 9).

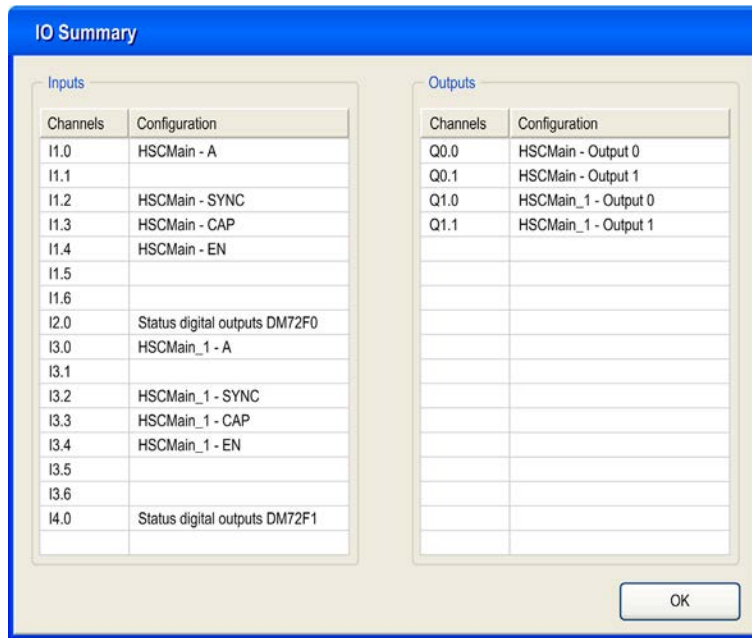
Configuration Procedure

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

Step	Action
1	Enter the Configuration Window .
2	Double click the controller.
3	Select Expert I/O → DM72F0/DM72F1 → HSCSimple Result: The HSC Main configuration window opens.
4	Set the mode to Free-large from the drop down menu, by selecting HSC Main configuration → Mode → Free-large
5	Set the preset value from Preset Condition → Preset condition Provide a value to the preset field.
6	Set the anti-bounce filtering value from the drop down menu, by selecting Input Usage → A (respectively B).
7	Optionally, select the SYNC (I2) , CAP (I3) , EN (I4) and REF (I5) check box and provides a value from the associated drop down menu to enable the Synchronization function (<i>see page 174</i>), Enable function (<i>see page 176</i>) and Capture function (<i>see page 162</i>) with a physical input.
8	Optionally, select the TH0 check box to provide a threshold value. This enables the Compare function and reflex outputs can be configured (<i>see page 153</i>).
9	The Limits defines the behavior of the counter when limits are reached.

IO Summary

The input/output configuration is displayed in the IO Summary window, opened with the **Summary** button:



Refer to the hardware guide for wiring details. (see *Modicon M258, Logic Controller, Hardware Guide*)

Programmable Filter

The filtering value on the **Main** type input determines the counter maximum frequency as shown in the table below:

Input	Filter value	Maximum counter frequency
A, B	0.002 ms	200 kHz
	0.004 ms	100 kHz
	0.012 ms	40 kHz
	0.04 ms	10 kHz
	0.12 ms	4 kHz
	0.4 ms	1 kHz
	1.2 ms	400 Hz
	4 ms	100 Hz


Programming the Main Type

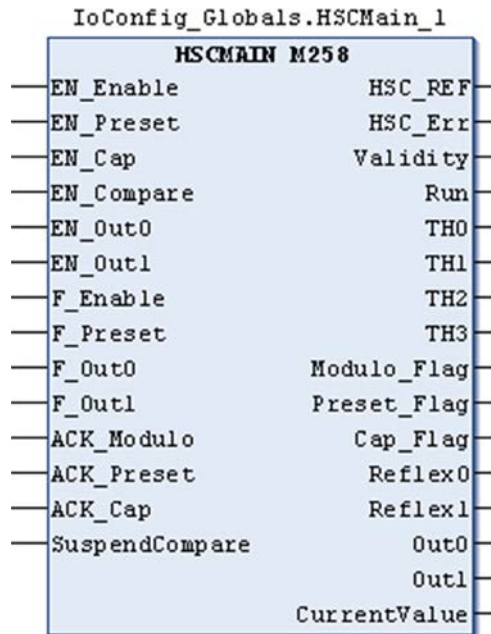
Overview

Main type is always managed by an HSCMain function block.

NOTE: At build, an error code is given if the HSCMain function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Insert the HSCMain_M258 function block with the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP → HSC → HSCMain_M258
2	Type the Main type instance name (defined in configuration, step 5) or look for the function block instance by clicking on:  Using the input assistant, the HSC instance can be selected at the following path: Global Variables → <MyController> → PLC Logic → IoConfig_Globals



I/O Variables Usage

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

The following table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When EN input is configured: if TRUE, authorizes the counter enable via the Enable input (<i>see page 176</i>).
EN_Preset	BOOL	When SYNC input is configured: if TRUE, authorizes the counter synchronization and start via the Sync input (<i>see page 172</i>).
EN_Cap	BOOL	When CAP input is configured: if TRUE, enables the Capture input (<i>see page 162</i>).
EN_Compare	BOOL	TRUE = enables the comparator operation (<i>see page 153</i>) (using Thresholds 0, 1, 2, 3): <ul style="list-style-type: none"> ● basic comparison (TH0, TH1, TH2, TH3 output bits) ● reflex (Reflex0, Reflex1 output bits) ● events (to trigger external tasks on threshold crossing)
EN_Out0	BOOL	TRUE = enables physical output Output0 to echo the Reflex0 value (if configured).
EN_Out1	BOOL	TRUE = enables physical output Output1 to echo the Reflex1 value (if configured).
F_Enable	BOOL	Forces the Enable condition (<i>see page 176</i>).
F_Preset	BOOL	Forces the Preset condition (<i>see page 172</i>).
F_Out0	BOOL	TRUE = forces Output0 to 1 (if Reflex0 is configured).
F_Out1	BOOL	TRUE = forces Output1 to 1 (if Reflex1 is configured).
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: <ul style="list-style-type: none"> ● TH0, TH1, TH2, TH3, Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value. ● Physical outputs Output0 and Output1 maintain their last value ● Events are masked <p>NOTE: EN_Compare, EN_Out0, EN_Out1, F_Out0, F_Out1 remain operational while SuspendCompare is set.</p>

The following table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF (see page 187)	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the Administrative function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (see page 193) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	Not relevant
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 (see page 153).
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 (see page 153).
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (see page 153).
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (see page 153).
Modulo_Flag	BOOL	Set to 1 when the counter rollovers its limits
Preset_Flag	BOOL	Set to 1 by the preset of the counter (see page 172)
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register (see page 162). This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0. (see page 153)
Reflex1	BOOL	State of Reflex1. (see page 153)
Out0	BOOL	State of physical outputs Output0 (if Reflex0 configured).
Out1	BOOL	State of physical outputs Output1 (if Reflex1 configured).
CurrentValue	DINT	Current counter value of the counter.

Adjusting Parameters

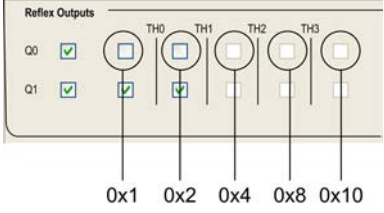
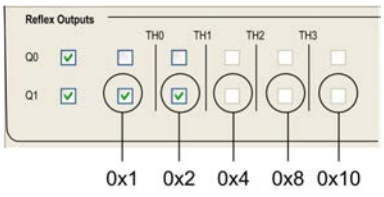
Overview

The list of parameters described in the table below can be read or modified by using the `EXPERTGetParam` (see page 196) or `EXPERTSetParam` (see page 198) function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M258 Logic Controller, Programming Guide*).

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 186) which can be read or modified while the program is running:

Parameter	Description
<code>EXPERT_PRESET</code>	to get or set the Preset value of the HSC
<code>EXPERT_THRESHOLD0</code>	to get or set the Threshold 0 value of an HSC
<code>EXPERT_THRESHOLD1</code>	to get or set the Threshold 1 value of an HSC
<code>EXPERT_THRESHOLD2</code>	to get or set the Threshold 2 value of an HSC
<code>EXPERT_THRESHOLD3</code>	to get or set the Threshold 3 value of an HSC
<code>EXPERT_REFLEX0</code>	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: 
<code>EXPERT_REFLEX1</code>	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: 

Event Counting Mode



Overview

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

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
12	Event Counting Principle	97
13	Event Counting With a Main Type	99

Event Counting Principle

12

Event Counting Mode Principle Description

Overview

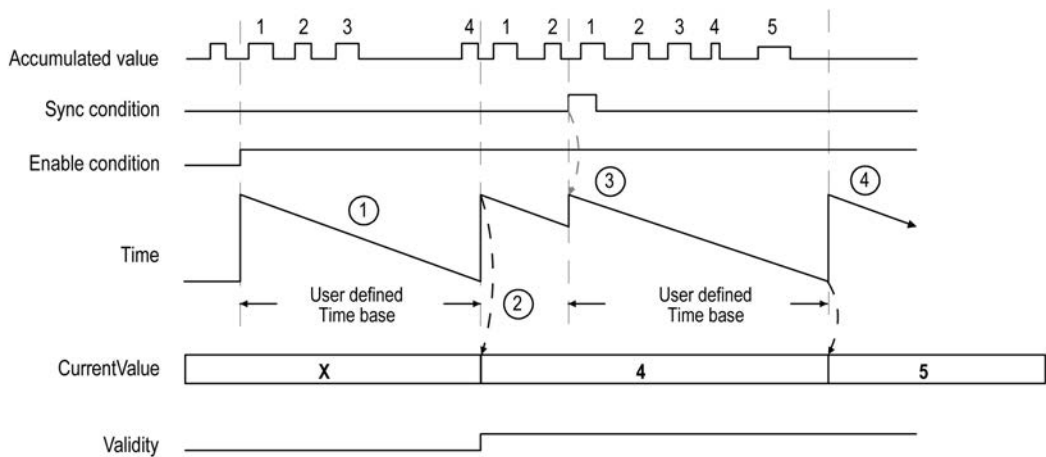
The **Event Counting** mode allows you to count a sequence of events during a given period of time.

Principle

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

The 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 172).

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 current 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: <ul style="list-style-type: none">● the accumulated value is reset to 0● the current 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 to 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.

Event Counting With a Main Type

13

Overview

This chapter describes how to implement a high speed counter in **Event Counting** mode using a **Main** type.

What's in this Chapter?

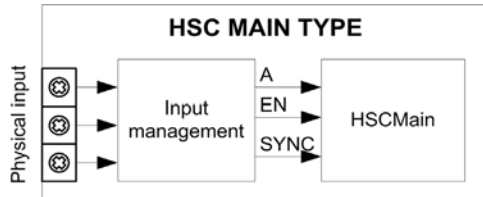
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	100
Configuration of the Main Type in Event Counting Mode	101
Programming the Main Type	104
Adjusting Parameters	107

Synopsis Diagram

Synopsis Diagram

The following diagram provides an overview of the **Main** type in **Event Counting** 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.

Optional Function

In addition to the **Event Counting** mode, the **Main** type provides the following functions:

- Synchronize by a physical input (*see page 172*)
- Enable by a physical input (*see page 176*)

Configuration of the Main Type in Event Counting Mode

Configuration Window

The following figure shows the **Main** type in **Event Counting** mode configuration window.

The numbers in the bullets are associated with the Configuration Procedure table:

Configuration Procedure

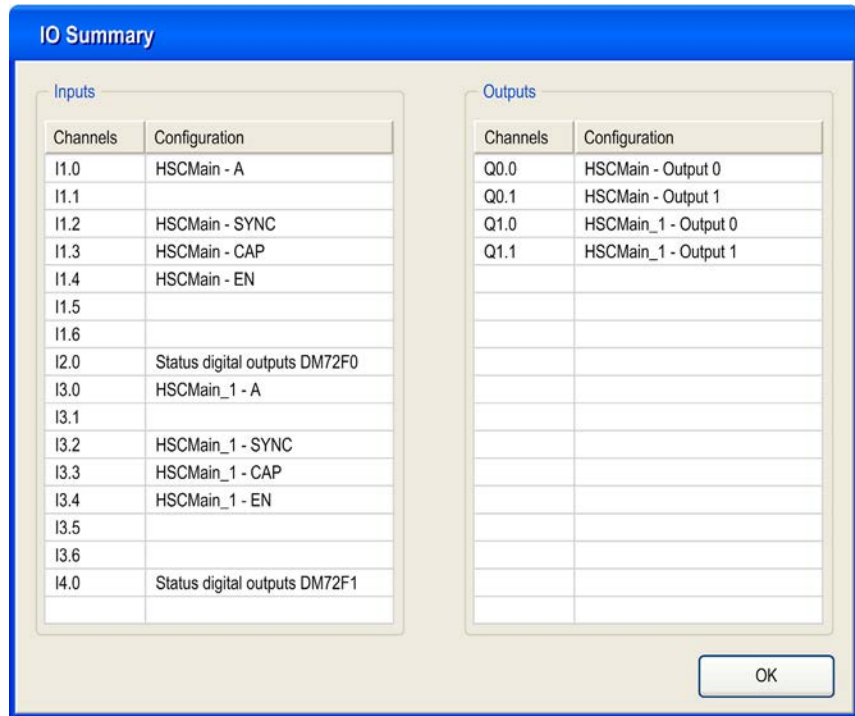
Follow this procedure to configure a **Main** type in **Event Counting** mode:

Step	Action
1	Enter the Configuration Window
2	Double click the controller
3	Select Expert I/O → DM72F0/DM72F1 → HSCMain Result: The HSC Main configuration window opens.
4	Set the mode to Event Counting from the drop down menu, by selecting HSC Main configuration → Mode → Event
5	Set the anti-bounce filtering value from the drop down menu, by selecting Filter → A

Step	Action
6	Optionally, select the SYNC (I2) and EN (I4) check box and provides a value from the associated drop down menu to enable the Synchronization function (see page 172) and Enable function (see page 176) with a physical input.
7	Set the preset condition value from the drop down menu, by selecting Preset Condition → Preset Condition
8	Set the time base value from the drop down menu, by selecting Time Base → Time base

IO Summary

The input/output configuration is displayed in the IO Summary window, opened with the **Summary** button:



Refer to the hardware guide for wiring details. (see *Modicon M258, Logic Controller, Hardware Guide*)

Programmable Filter

The filtering value on the **Main** type input determines the counter maximum frequency as shown in the table:

Input	Filter value	Maximum counter frequency
A	0.002 ms	200 kHz
	0.004 ms	100 kHz
	0.012 ms	40 kHz
	0.04 ms	10 kHz
	0.12 ms	4 kHz
	0.4 ms	1 kHz
	1.2 ms	400 Hz
	4 ms	100 Hz


Programming the Main Type

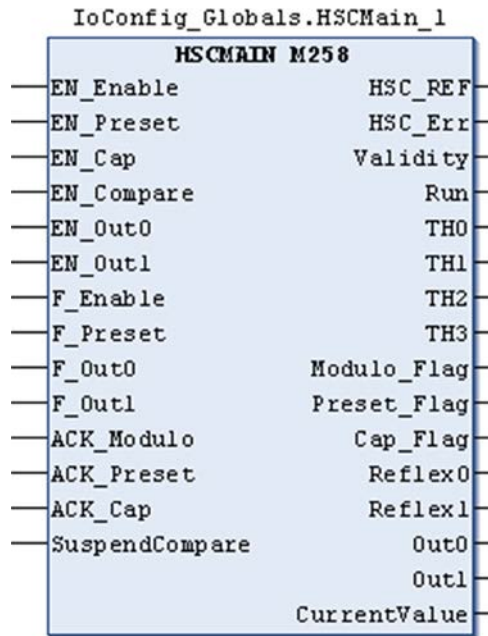
Overview

Main type is always managed by an HSCMain function block.

NOTE: At build, an error code is given if the HSCMain function block is used to manage a different HSC type.

Adding the HSCMain Function Block

Step	Description
1	Insert the HSCMain_M258 function block with the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP → HSC → HSCMain_M258
2	Type the Main type instance name (defined in configuration, step 5) or look for the function block instance by clicking on:  Using the input assistant, the HSC instance can be selected at the following path: Global Variables → <MyController> → PLC Logic → IoConfig_Globals



I/O Variables Usage

This table describes how the different pins of the function block are used in the mode **Event**.

The following table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When EN input is configured: if TRUE, authorizes the counter enable via the Enable input (<i>see page 176</i>).
EN_Preset	BOOL	When SYNC input is configured: if TRUE, authorizes the counter synchronization and start via the Sync input (<i>see page 172</i>).
EN_Cap	BOOL	Not used
EN_Compare	BOOL	Not used
EN_Out0	BOOL	Not used
EN_Out1	BOOL	Not used
F_Enable	BOOL	Forces the Enable condition (<i>see page 176</i>).
F_Preset	BOOL	Forces the Preset condition (<i>see page 172</i>).
F_Out0	BOOL	Not used
F_Out1	BOOL	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

The following table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF (<i>see page 187</i>)	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the Administrative function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. EXPERTGetDiag (<i>see page 193</i>) function block may be used to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	Not relevant
TH0	BOOL	Not relevant
TH1	BOOL	Not relevant
TH2	BOOL	Not relevant

Outputs	Type	Comment
TH3	BOOL	Not relevant
Modulo_Flag	BOOL	Not relevant
Preset_Flag	BOOL	Set to 1 by the preset of the counter (see page 172)
Cap_Flag	BOOL	Not relevant
Reflex0	BOOL	Not relevant
Reflex1	BOOL	Not relevant
Out0	BOOL	Not relevant
Out1	BOOL	Not relevant
CurrentValue	DINT	Current counter value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table below can be read or modified by using the `EXPERTGetParam` (see page 196) or `EXPERTSetParam` (see page 198) function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M258 Logic Controller, Programming Guide*).

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 186) which can be read or modified while the program is running:

Parameter	Description
<code>EXPERT_TIMEBASE</code>	to get or set the Timebase value of the HSC

Frequency Meter Mode

A large, bold, black Roman numeral 'VI' is centered within a light gray square background.

Overview

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

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
14	Frequency Meter Principle	111
15	Frequency Meter Mode With a Main Type	113

Frequency Meter Principle

14

Frequency Meter Mode Principle Description

Overview

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

The **Main** type in **Frequency meter** mode calculates the number of pulses in time intervals of 1 s. An updated value in Hz is available every 10 ms.

When there is a variation in 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 100 kHz. Beyond 100 kHz, the counting register value may decrease until it reaches 0. Beyond 100 kHz and up to the real cut-off frequency of 100 kHz, the **Main** type may indicate that it has exceeded the frequency limit.

The maximum duty cycle at 100 kHz is 60%.

Frequency Meter Mode With a Main Type

15

Overview

This chapter describes how to implement a high speed counter in **Frequency meter** mode using a **Main** type.

What's in this Chapter?

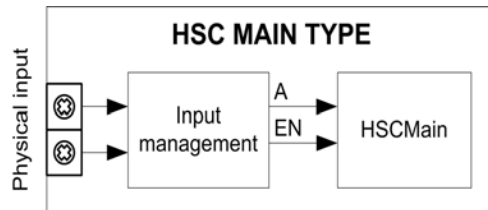
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	114
Configuration of the Main Type in Frequency Meter Mode	115
Programming the Main Type	117

Synopsis Diagram

Synopsis Diagram

The following diagram provides an overview of the **Main** type in **Frequency meter** mode:



A is the counting input of the counter.

EN is the enable input of the counter.

Optional Function

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

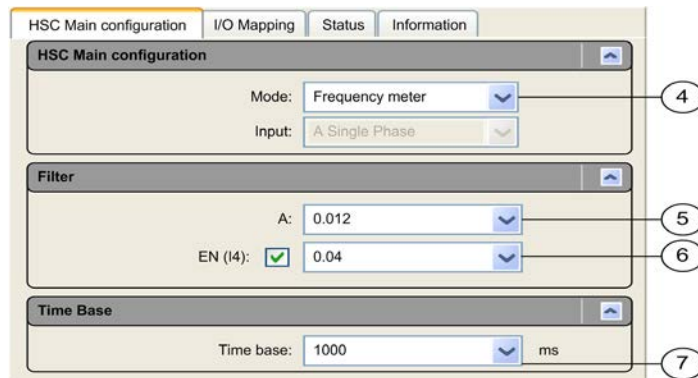
- Enable by a physical input (*see page 176*)

Configuration of the Main Type in Frequency Meter Mode

Configuration Window

The following figure represents the **Main** type in **Frequency meter** mode configuration window.

The numbers in the bullets are associated with the Configuration Procedure table:



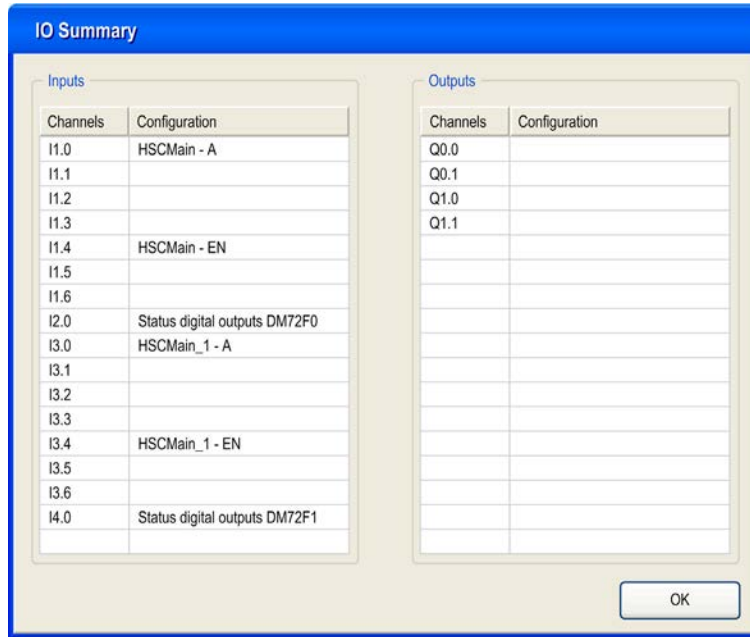
Configuration Procedure

Follow this procedure to configure a **Main** type in **Frequency meter** mode:

Step	Action
1	Enter the Configuration Window
2	Double click the controller
3	Select Expert I/O → DM72F0/DM72F1 → HSCMain Result: The HSC Main configuration window opens.
4	Set the mode to Frequency meter from the drop down menu, by selecting HSC Main configuration → Mode → Frequency meter
5	Set the anti-bounce filter value from the drop down menu, by selecting Filter → A
6	Optionally, select the EN (I4) check box and provide a value from the associated drop down menu to enable the Enable function (see page 176) with a physical input.
7	Set the time base value from the drop down menu, by selecting Time Base → Time Base

IO Summary

The input/output configuration is displayed in the IO Summary window, opened with the **Summary** button:



Refer to the hardware guide for wiring details. (see *Modicon M258, Logic Controller, Hardware Guide*)

Programmable Filter

The filtering value on the **Main** type input determines the counter maximum frequency as shown in the table:

Input	Filter value	Maximum counter frequency
A	0.002 ms	200 kHz
	0.004 ms	100 kHz
	0.012 ms	40 kHz
	0.04 ms	10 kHz
	0.12 ms	4 kHz
	0.4 ms	1 kHz
	1.2 ms	400 Hz
	4 ms	100 Hz


Programming the Main Type

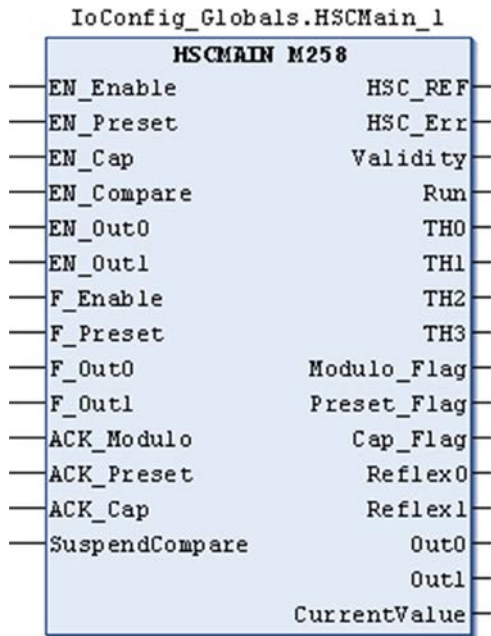
Overview

Main type is always managed by an HSCMain function block.

NOTE: At build, an error code is given if the HSCMain function block is used to manage a different HSC type.

Adding a HSCMain Function Block

Step	Description
1	Insert the HSCMain_M258 function block with the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP →HSC →HSCMain_M258
2	Type the Main type instance name (defined in configuration, step 5) or look for the function block instance by clicking on:  Using the input assistant, the HSC instance can be selected at the following path: Global Variables →<MyController> →PLC Logic → IoConfig_Globals



I/O Variables Usage

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

The following table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When EN input is configured: if TRUE, authorizes the counter enable via the Enable input (<i>see page 176</i>).
EN_Preset	BOOL	Not used
EN_Cap	BOOL	Not used
EN_Compare	BOOL	Not used
EN_Out0	BOOL	Not used
EN_Out1	BOOL	Not used
F_Enable	BOOL	Forces the Enable condition (<i>see page 176</i>).
F_Preset	BOOL	Forces the Preset condition (<i>see page 172</i>)
F_Out0	BOOL	Not used
F_Out1	BOOL	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

The following table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF (<i>see page 187</i>)	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the Administrative function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (<i>see page 193</i>) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	Not relevant
TH0	BOOL	Not relevant
TH1	BOOL	Not relevant
TH2	BOOL	Not relevant
TH3	BOOL	Not relevant

Outputs	Type	Comment
Modulo_Flag	BOOL	Not relevant
Preset_Flag	BOOL	Set to 1 by the preset of the counter (<i>see page 172</i>)
Cap_Flag	BOOL	Not relevant
Reflex0	BOOL	Not relevant
Reflex1	BOOL	Not relevant
Out0	BOOL	Not relevant
Out1	BOOL	Not relevant
CurrentValue	DINT	Current counter value of the counter.

Period Meter Mode



Overview

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

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
16	Period Meter Mode Principle	123
17	Period Meter With a Main Type	125

Period Meter Mode Principle

16

Period Meter Mode Principle Description

Overview

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

The **Period meter** can be used in 2 ways:

- Edge to opposite: Allows the measure of the duration of an event.
- Edge to edge: Allows the measure of the length of time between 2 events.

A time out value can be specified in the configuration screen.

This function allows to stop a measurement that exceeds this time out. In this case, the counting register is not valid until the next complete measurement.

The measure is expressed in the unit defined by the resolution parameter (1 μ s, 100 μ s, 1000 μ s).

Example, if the `CurrentValue` = 100 and the **Resolution** parameter is:

0.001 (1 μ s) measure = 0.1 ms

0.1 (100 μ s) measure = 10 ms

1 (1000 μ s) measure = 100 ms

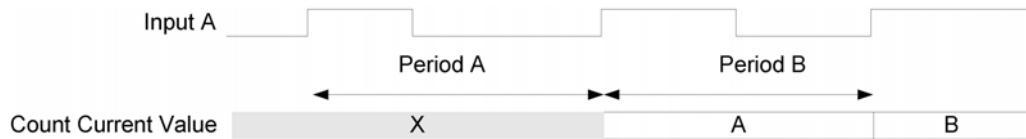
Edge to Opposite Mode

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.



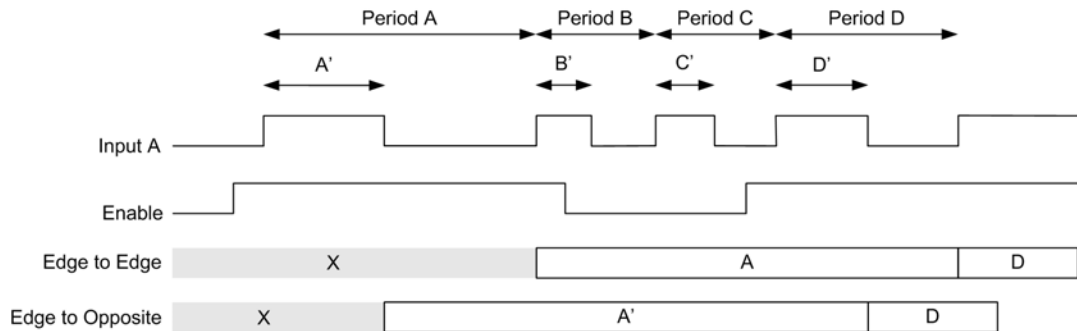
Edge to Edge Mode

When the Enable condition = 1, the measurement is taken between 2 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:



Operating Limits

The module can perform a maximum of 1 measurement every 5 ms.

The shortest pulse that can be measured is 100 μ s, even if the unit defined in the configuration is 1 μ s.

The maximum duration that can be measured is 1,073,741,823 units.

Period Meter With a Main Type

17

Overview

This chapter describes how to implement a high speed counter in **Period meter** mode using a **Main** type.

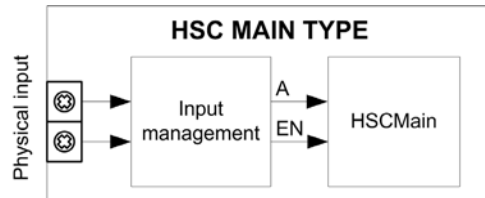
What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Synopsis Diagram	126
Configuration of the Main Type in Period Meter Mode	127
Programming the Main Type	130

Synopsis Diagram

Synopsis Diagram



A is the counting input of the counter.

EN is the enable input of the counter.

Optional Function

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

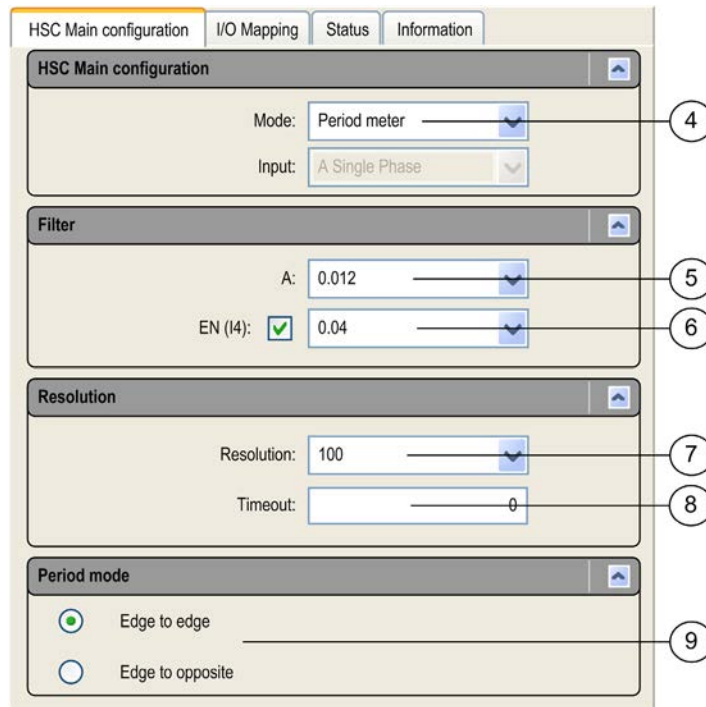
- Enable by a physical input (*see page 176*)

Configuration of the Main Type in Period Meter Mode

Configuration Window

The following figure represents the **Main** type in **Period meter** mode configuration window:

The numbers in the bullets are associated with the Configuration Procedure table:



Configuration Window

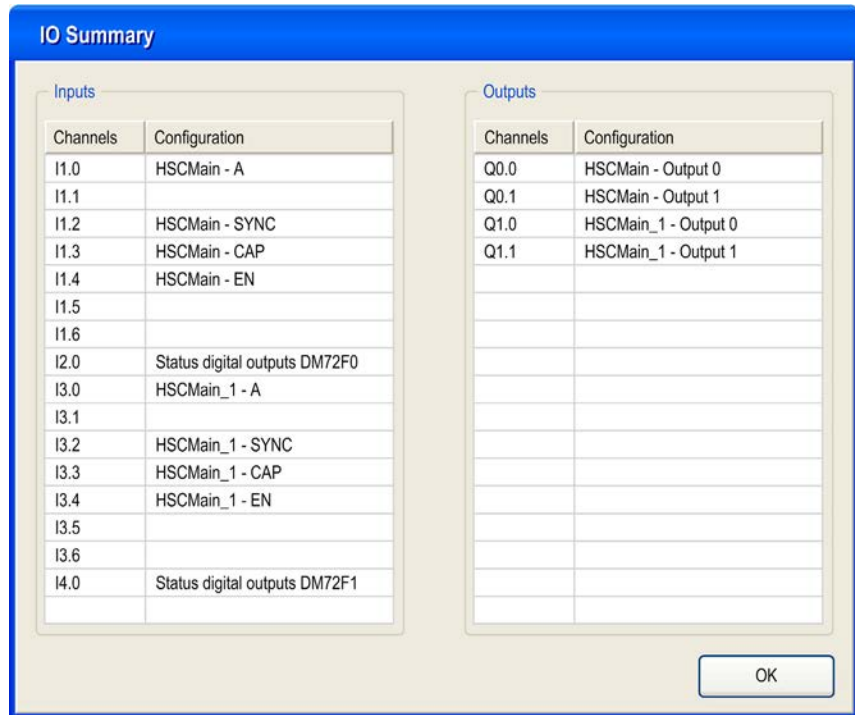
Follow this procedure to configure a **Main** type in **Period meter** mode:

Step	Action
1	Enter the Configuration Window .
2	Double click the controller.
3	Select Expert I/O → DM72F0/DM72F1 → HSCMain Result: The HSC Main configuration window opens.
4	Set the mode to Period meter from the drop down menu, by selecting HSC Main configuration → Mode → Period meter

Step	Action
5	Set the anti-bounce filters value from the drop down menu, by selecting Filter → A
6	Optionally, select the EN (I4) check box and provides a value from the associated drop down menu to enable the Enable function (see page 176) with a physical input.
7	Set the resolution value from the drop down menu, by selecting Resolution → Resolution
8	Set the time-out value from Resolution → Time-out
9	Set the period meter mode.

IO Summary

The input/output configuration is displayed in the IO Summary window, opened with the **Summary** button:



Refer to the hardware guide for wiring details. (see *Modicon M258, Logic Controller, Hardware Guide*)

Programmable Filter

The filtering value on the **Main** type input determines the counter maximum frequency as shown in the table:

Input	Filter value	Maximum counter frequency
A	0.002 ms	200 kHz
	0.004 ms	100 kHz
	0.012 ms	40 kHz
	0.04 ms	10 kHz
	0.12 ms	4 kHz
	0.4 ms	1 kHz
	1.2 ms	400 Hz
	4 ms	100 Hz


Programming the Main Type

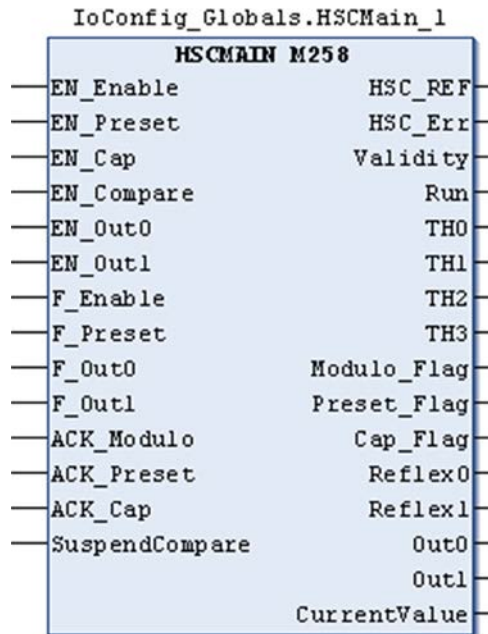
Overview

An HSC Main module is always managed by a HSCMain function block.

NOTE: An error code at compilation is given if the HSCMain function block is used to manage a different HSC type.

Adding a HSCMain Function Block

Step	Description
1	Insert the HSCMain_M258 function block with the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP → HSC → HSCMain_M258
2	Type the Main type instance name or look for the function block instance by clicking on:  Using the input assistant, the HSC instance can be selected at the following path: Global Variables → <MyController> → PLC Logic → IoConfig_Globals



I/O Variables Usage

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

The following table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When EN input is configured: if TRUE , authorizes the counter enable via the Enable input (<i>see page 176</i>).
EN_Preset	BOOL	Not used
EN_Cap	BOOL	Not used
EN_Compare	BOOL	Not used
EN_Out0	BOOL	Not used
EN_Out1	BOOL	Not used
F_Enable	BOOL	Forces the Enable condition (<i>see page 176</i>).
F_Preset	BOOL	Not used
F_Out0	BOOL	Not used
F_Out1	BOOL	Not used
ACK_Modulo	BOOL	Not used
ACK_Preset	BOOL	Not used
ACK_Cap	BOOL	Not used
SuspendCompare	BOOL	Not used

The following table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF (<i>see page 187</i>)	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the Administrative function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (<i>see page 193</i>) function block used to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid. If the time out value is exceeded, Validity = FALSE .
Run	BOOL	TRUE = Counter is running.
TH0	BOOL	Not relevant
TH1	BOOL	Not relevant
TH2	BOOL	Not relevant
TH3	BOOL	Not relevant

Outputs	Type	Comment
Modulo_Flag	BOOL	Not relevant
Preset_Flag	BOOL	Not relevant.
Cap_Flag	BOOL	Not relevant
Reflex0	BOOL	Not relevant
Reflex1	BOOL	Not relevant
Out0	BOOL	Not relevant
Out1	BOOL	Not relevant
CurrentValue	DINT	Current counter value of the counter.

Encoder



VIII

Incremental Mode With an Encoder

18

Overview

This chapter describes how to use an **Encoder** in **Incremental** mode.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Incremental Mode Principle Description	136
Synopsis Diagram	140
Configuration of the Standard Encoder on an Expert I/O Module	141
Programming the Standard Encoder	146
Adjusting Parameters	149

Incremental Mode Principle Description

Overview

Use of the **Incremental** mode to connect incremental encoders.

Principle

The **Incremental** mode behaves like a standard up/down counter.

When the Enable condition (*see page 176*) is false, the counter ignores the pulses applied to the counting inputs A/B.

In the **Incremental** mode, the counter must be preset at least one time to operate. The current counter value is loaded with the `Preset` value each time the Preset condition (*see page 172*) occurs.

The current counter can be stored in the capture register by configuring the Capture conditions (*see page 165*).

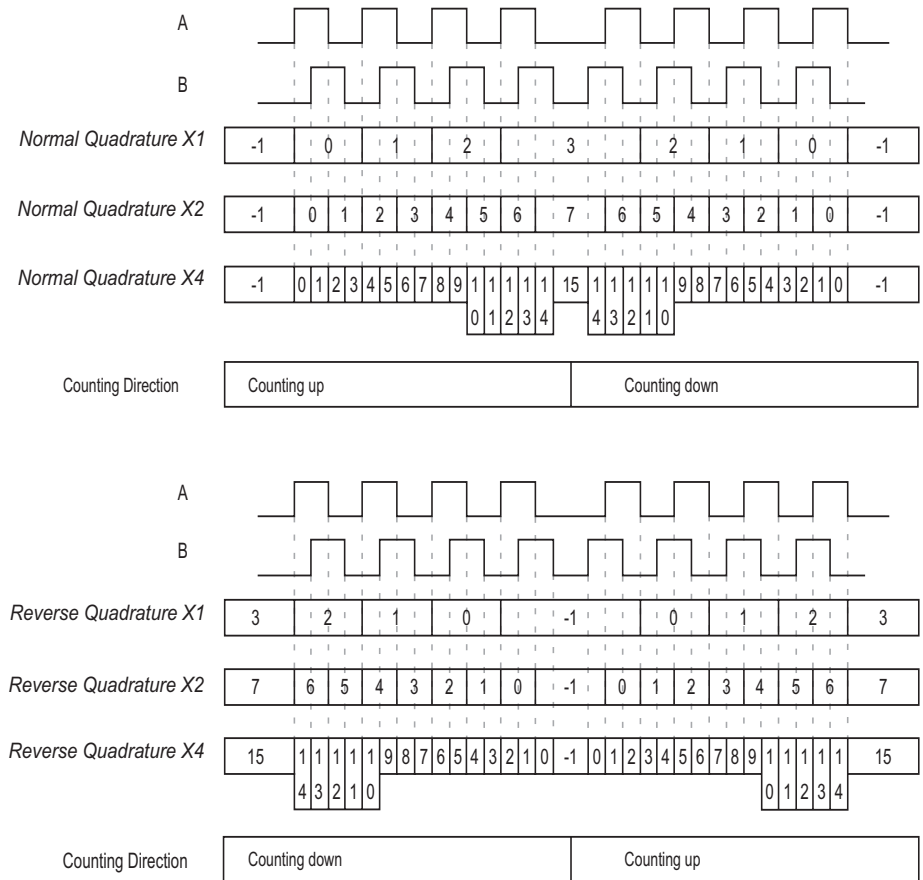
Axis Types

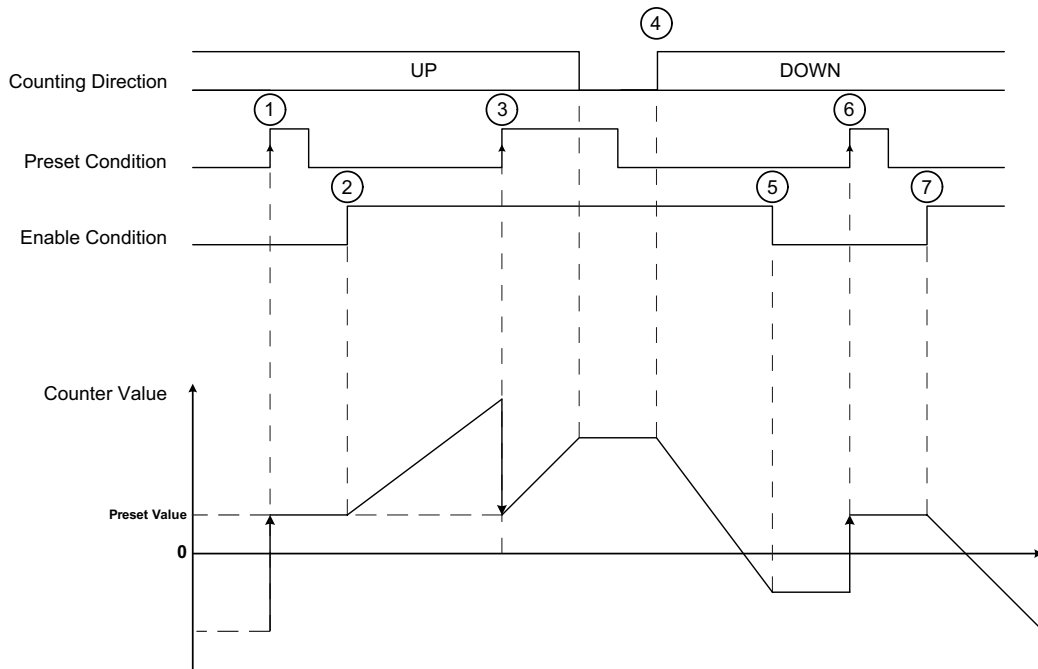
The following table shows the 2 available axis types:

Axis Type	Comment
Linear	This mode acts as a finite counter.
Rotary	This mode acts as an infinite counter.

Principle Diagram

The input mode in **Incremental** mode is always a quadrature:





Stage	Action
1	On the rising edge of Preset condition, the current value is set to the preset value and the counter is activated.
2	When the Enable condition = 1, the counter starts to increment when the counting direction is up.
3	The rising edge on the Preset condition loads the Preset value.
4	When the Enable condition = 1, the counter starts to decrement when the counting direction is down.
5	When the Enable condition = 0, the counter ignores the pulses applied to the counting inputs A/B.
6	The rising edge on the Preset condition loads the preset value.
7	When the Enable condition = 1, the counter starts to decrement when the counting direction is down.

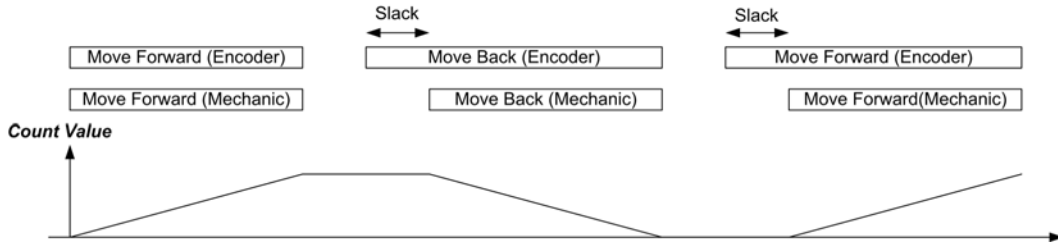
NOTE: Enable and Preset conditions depends on configuration. These are described in the Enable (see page 176) and Preset (see page 174) function.

Slack

The counter applies an hysteresis if the rotation is inverted. The value of slack defines the number of points that are not acknowledged by the counter during the rotation inversion.

This takes into account the slack between the encoder/motor axis and the mechanical axis (e.g. an encoder measuring the position of a mat).

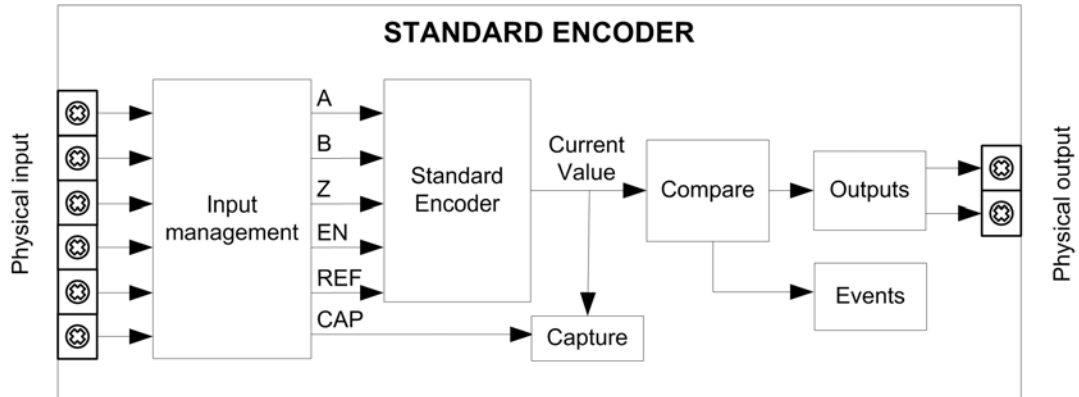
This behavior is illustrated in the following figure:



Synopsis Diagram

Synopsis Diagram

The following diagram provides an overview of the **Standard Encoder** in **Incremental** mode:



A and B are the counting inputs of the encoder.

EN is the enable input of the encoder.

Z and REF are the reference inputs of the encoder.

CAP is the capture input of the encoder.

Optional Function

In addition to the **Incremental** mode, the **Standard Encoder** can provide the following function:

- Compare (see page 153)
- Capture (see page 165)
- Enable with a physical input (see page 176)
- Preset a physical input (see page 172)

Configuration of the Standard Encoder on an Expert I/O Module

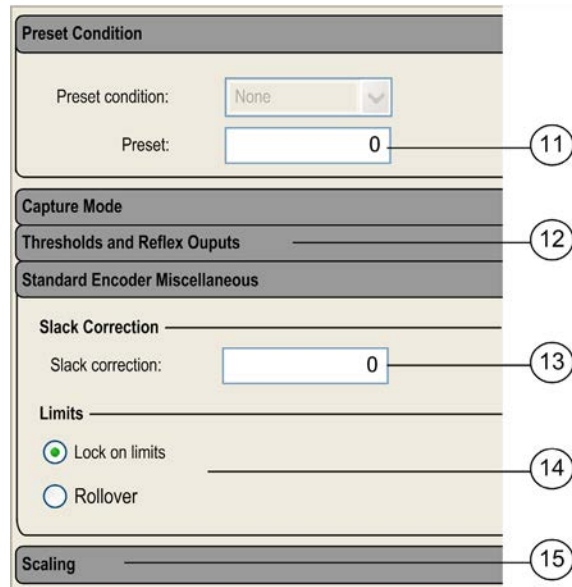
Configuration Window

The following figure shows the **Standard Encoder** type in **Incremental** mode configuration window.

The numbers in the bullets are associated with the Configuration Procedure table:

The image shows a configuration window for a Standard Encoder. It is divided into two main sections: 'Encoder common configuration' and 'Input Usage'. The 'Encoder common configuration' section includes three dropdown menus: 'Mode' set to 'Incremental', 'Input' set to 'Normal Quadrature X1', and 'Axis Type' set to 'Linear'. The 'Input Usage' section lists six input channels (I0 through I5). Each channel has a status dropdown (set to 'Disabled') and a pulse width dropdown (set to '0.002 ms'). Numbered callouts (4-10) point to specific elements: 4 points to the 'Input' dropdown, 5 to the 'Axis Type' dropdown, 6 to the pulse width dropdowns for I0 and I1, 7 to the status dropdown for I2, 8 to the status dropdown for I3, 9 to the status dropdown for I4, and 10 to the status dropdown for I5.

Section	Parameter	Value	Unit	Callout	
Encoder common configuration	Mode	Incremental			
	Input	Normal Quadrature X1		4	
	Axis Type	Linear		5	
Input Usage	I0 (A)	0.002	ms	6	
	I1 (B)	0.002	ms		
	I2	Disabled	0.002	ms	7
	I3	Disabled	0.002	ms	8
	I4	Disabled	0.002	ms	9
	I5	Disabled	0.002	ms	10



Configuration Procedure

Follow this procedure to configure a **Standard Encoder** in **Incremental Mode**:

Step	Action
1	Enter the Configuration Window
2	Double click the controller
3	Select Expert I/O → DM72F0/DM72F1 → Standard Encoder Result: The Standard Encoder configuration window opens
4	Set the input mode from the Input drop down menu.
5	Select the type of axis in the Axis drop down menu: <ul style="list-style-type: none"> ● Linear ● Rotary
6	Select the anti-bounce filtering value on A in the A Filter list box. Select the anti-bounce filtering value on B in the B Filter list box.
7	If the physical input Z is used, select Z from the I2 drop menu and configure the associated anti-bounce filter.
8	If the physical input CAP is used, select CAP from the I3 drop menu and configure the associated anti-bounce filter.
9	If the physical input EN is used, select EN from the I4 drop menu and configure the associated anti-bounce filter.

Step	Action												
10	If the physical input REF is used, select REF from the I5 drop menu and configure the associated anti-bounce filter.												
11	Specify a preset value in the Preset condition area. If available, select the Preset condition logic.												
12	Configure the comparison function (<i>see page 153</i>).												
13	Specify the slack correction value.												
14	Specify the Limits (<i>see page 83</i>) behavior of the encoder (only for Linear axis).												
15	<p>Scaling is the conversion between counter value and increments. If the Axis is a Linear axis:</p> <div data-bbox="473 511 889 690" style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p>Scaling</p> <p>Increments / Units</p> <p>Increments: <input type="text" value="15"/></p> <p>Units: <input type="text" value="10"/></p> </div> <p>Encoder Value = (Units / Increments x counting pulses. Example: If a conveyor moves 10 cm on 15 counting pulses, Units = 10 and Increments = 15. Every 15 pulses, the encoder value increases by 10.</p> <p>If your Axis is a Rotary axis:</p> <div data-bbox="473 852 902 1242" style="border: 1px solid black; padding: 5px; margin-bottom: 10px;"> <p>Scaling</p> <p>Increments / Units</p> <p>Increments: <input type="text" value="15"/></p> <p>Units: <input type="text" value="10"/></p> <p>Offset / Modulo</p> <p>Offset: <input type="text" value="0"/></p> <p>Modulo: <input type="text" value="100"/></p> </div> <p>(Modulo x Increment)/Units = Integer If this rule is not respected, there will be a slip.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Modulo</th> <th>Offset</th> <th>The encoder counts</th> </tr> </thead> <tbody> <tr> <td>100</td> <td>0</td> <td>from 0 to 99</td> </tr> <tr> <td>100</td> <td>20</td> <td>from 20 to 119</td> </tr> <tr> <td>100</td> <td>-20</td> <td>from -20 to 79</td> </tr> </tbody> </table>	Modulo	Offset	The encoder counts	100	0	from 0 to 99	100	20	from 20 to 119	100	-20	from -20 to 79
Modulo	Offset	The encoder counts											
100	0	from 0 to 99											
100	20	from 20 to 119											
100	-20	from -20 to 79											

⚠ WARNING

Inaccurate Encoder Value

Respect the mathematical rule of modulo $((\text{Modulo} \times \text{Increment})/\text{Unit} = \text{Integer})$ to avoid slipping.

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

IO Summary

The input/output configuration is displayed in the IO Summary window (see page 23):

The screenshot shows a software window titled "IO Summary" with a blue header. It contains two main sections: "Inputs" and "Outputs", each with a table of channel configurations. An "OK" button is located at the bottom right of the window.

Inputs		Outputs	
Channels	Configuration	Channels	Configuration
I1.0	Standard_Encoder - A	Q0.0	Standard_Encoder - Output 0
I1.1	Standard_Encoder - B	Q0.1	Standard_Encoder - Output 1
I1.2	Standard_Encoder - Z	Q1.0	Standard_Encoder_1 - Output 0
I1.3	Standard_Encoder - REF/CAP1	Q1.1	Standard_Encoder_1 - Output 1
I1.4	Standard_Encoder - EN/CAP0		
I1.5	Standard_Encoder - REF		
I1.6			
I2.0	Status digital outputs DM72F0		
I3.0	Standard_Encoder_1 - A		
I3.1	Standard_Encoder_1 - B		
I3.2	Standard_Encoder_1 - Z		
I3.3	Standard_Encoder_1 - REF/CAP1		
I3.4	Standard_Encoder_1 - EN/CAP0		
I3.5	Standard_Encoder_1 - REF		
I3.6			
I4.0	Status digital outputs DM72F1		

Refer to the hardware guide for wiring details. (see *Modicon M258, Logic Controller, Hardware Guide*)

Programmable Filter

The filtering value on the **Standard Encoder** input determines the counter maximum frequency, as shown in the table below:


Input	Filter value	Maximum counter frequency
A, B	0.002 ms	200 kHz
	0.004 ms	100 kHz
	0.012 ms	40 kHz
	0.04 ms	10 kHz
	0.12 ms	4 kHz
	0.4 ms	1 kHz
	1.2 ms	400 Hz
	4 ms	100 Hz

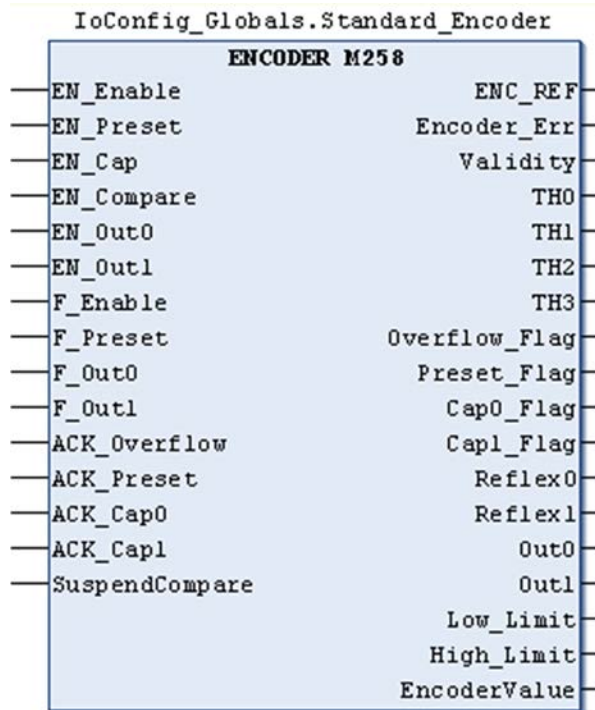
Programming the Standard Encoder

Overview

A **Standard Encoder** is always managed by an Encoder_M258 (see page 200) function block.

Adding a StandardEncoder Function Block

Step	Description
1	Insert the Encoder_M258 function block with the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP → ENCODER → ENCODER_M258
2	Type the Encoder_M258 instance name or look for the function block instance by clicking on:  Using the input assistant, the Encoder_M258 instance can be selected at the following path: Global Variables → PLC Logic → IoConfig_Globals



I/O Variables Usage

The following table describes the input variables:

Inputs	Type	Comment
EN_Enable	BOOL	When EN input is configured authorizes the encoder enable via the input (<i>see page 176</i>).
EN_Preset	BOOL	When Z or REF input are configured authorizes the counter preset via the inputs (<i>see page 172</i>).
EN_Cap	BOOL	When at least one CAP input is configured authorizes the capture function via the inputs (<i>see page 165</i>).
EN_Compare	BOOL	TRUE = enables the comparator operation using Thresholds 0, 1, 2, 3 (<i>see page 153</i>): <ul style="list-style-type: none"> ● basic comparison (TH0, TH1, TH2, TH3 output bits) ● reflex (Reflex0, Reflex1 output bits) ● events (to trigger external tasks on threshold crossing)
EN_Out0	BOOL	TRUE = authorizes physical output Output0 to echo the Reflex0 value.
EN_Out1	BOOL	TRUE = authorizes physical output Output1 to echo the Reflex1 value.
F_Enable	BOOL	Forces the Enable condition (<i>see page 176</i>).
F_Preset	BOOL	Forces the Preset condition.
F_Out0	BOOL	TRUE = forces physical output Output0 to 1 (if Reflex0 is configured) (<i>see page 153</i>).
F_Out1	BOOL	TRUE = forces physical output Output1 to 1 (if Reflex1 is configured) (<i>see page 153</i>).
ACK_Overflow	BOOL	On rising edge, resets Overflow_Flag
ACK_Preset	BOOL	On rising edge, resets Preset_Flag (<i>see page 172</i>).
ACK_Cap0	BOOL	On rising edge, resets Cap0_Flag (<i>see page 165</i>).
ACK_Cap1	BOOL	On rising edge, resets Cap1_Flag (<i>see page 165</i>).
SuspendCompare	BOOL	TRUE = the comparator operation results are frozen (<i>see page 153</i>): <ul style="list-style-type: none"> ● TH0, TH1, TH2, TH3, Reflex0, Reflex1 output bits maintain their last value. ● Physical outputs Output0 and Output1 maintain their last value. ● Events are masked. <p>EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.</p>

The following table describes the output variables:

Outputs	Type	Comment
ENC_REF	EXPERT_REF (see page 187)	Reference to the Standard Encoder. To be used with the EXPERT_REF_IN input of Administrative Function block
Encoder_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (see page 193) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that the output values on the function block are valid. TRUE after the first preset
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 (if configured) (see page 153).
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 (if configured) (see page 153).
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (if configured) (see page 153).
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (if configured) (see page 153).
Overflow_Flag	BOOL	Set to 1 when the encoder rolls over its limits.
Preset_Flag	BOOL	Set to 1 after the encoder presets (see page 174).
Cap0_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register (see page 165). This flag must be reset before a new capture is allowed.
Cap1_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register (see page 165). This flag must be reset before a new capture is allowed.
Reflex0	BOOL	State of Reflex0 (see page 153).
Reflex1	BOOL	State of Reflex1 (see page 153).
Out0	BOOL	State of Output0 (see page 153).
Out1	BOOL	State of Output1 (see page 153).
Low_Limit	BOOL	Set to 1 when the encoder exceeds -2.147.483.648. (see page 83) Reset to 0 when encoder presets.
High_Limit	BOOL	Set to 1 when the encoder exceeds +2.147.483.647. (see page 83) Reset to 0 when encoder presets
EncoderValue	DINT	Current value of the encoder.

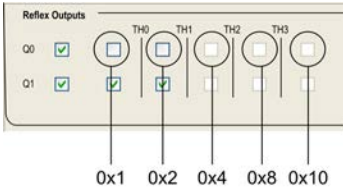
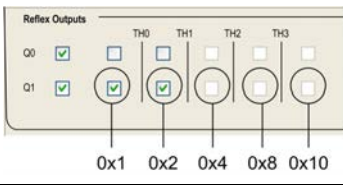
Adjusting Parameters

Overview

The list of parameters described in the table below can be read or modified by the using the `EXPERTGetParam` (see page 196) or `EXPERTSetParam` (see page 198) function blocks.

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 186) which can be modified while the program is running:

Parameter	Description
<code>EXPERT_PRESET</code>	to get or set the Preset value of the Encoder
<code>EXPERT_THRESHOLD0</code>	to get or set the Threshold 0 value of an Encoder
<code>EXPERT_THRESHOLD1</code>	to get or set the Threshold 1 value of an Encoder
<code>EXPERT_THRESHOLD2</code>	to get or set the Threshold 2 value of an Encoder
<code>EXPERT_THRESHOLD3</code>	to get or set the Threshold 3 value of an Encoder
<code>EXPERT_OFFSET</code>	to get or set OFFSET of an Encoder in Rotary axis mode
<code>EXPERT_SLACK</code>	to get or set Slack of an Encoder
<code>EXPERT_SCALING</code>	to get or set the Scaling parameter of an Encoder The scaling parameter (<code>ParamValue</code> of the function block) is made of 2 sub-parameters: High significant INT = Increments Low significant INT = Units
<code>EXPERT_REFLEX0</code>	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: 
<code>EXPERT_REFLEX1</code>	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: 

Optional Functions



Overview

This part provides information on optional function for HSC.

What's in this Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
19	Comparison Function	153
20	Capture Function	161
21	Preset and Enable Functions	171

Comparison Function

19

19.1 Comparison with a Main Type

Overview

This section provides information on the comparison function with a **Main** type or an encoder.

What's in this Section?

This section contains the following topics:

Topic	Page
Comparison Principle with a Main type or an Encoder	154
Configuration of the Comparison on a Main Type or an Encoder	158
External Event Configuration	159

Comparison Principle with a Main type or an Encoder

Overview

The compare block with the **Main** type or an Encoder manages Thresholds, Reflex outputs and Events in the following modes:

- One-shot (see page 37)
- Modulo-loop (see page 55)
- Free-Large (see page 77)

Comparison is configured in the Configuration screen (see page 158) by activating at least one threshold.

Comparison can be used to trigger:

- programming action on thresholds (see page 155)
- an event on threshold associated with an external task (see page 155)
- reflex outputs (see page 155)

Principle of a Comparison

The **Main** type or an Encoder can manage up to 4 thresholds.

A threshold is a configured value that is compared to the current counting value. Thresholds are used to define up to 5 zones or to react to a value crossing.

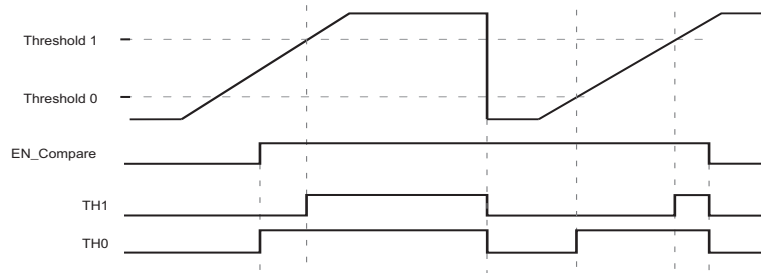
They are defined by configuration and can also be adjusted in the application program by using the EXPERTSetParam (see page 198) function block.

If Threshold_x (x= 0, 1, 2, 3) is configured and comparison is enabled (EN_Compare = 1), output pin TH_x of the HSCMain_M258 (Encoder_M258) function block is:

- set when counter value \geq Threshold_x
- reset when counter value $<$ threshold_x

NOTE: When EN_Compare is set to 0 on HSCMain_M258 (Encoder_M258) function block, comparison functions are disabled, including external tasks triggered by a threshold event and Reflex outputs.

Example for 2 thresholds:



Threshold Behavior

Using thresholds comparison status available in the task context (TH0 to TH2 output pins of the function block) is suitable for an application with a low time constant.

It can be used, for example, to monitor the liquid level in a tank.

Configuring Event

Configuring an event on threshold crossing allows to trigger an external task (see page 159). You can choose to trigger an event when a configured threshold is crossed downward, upward, or both.

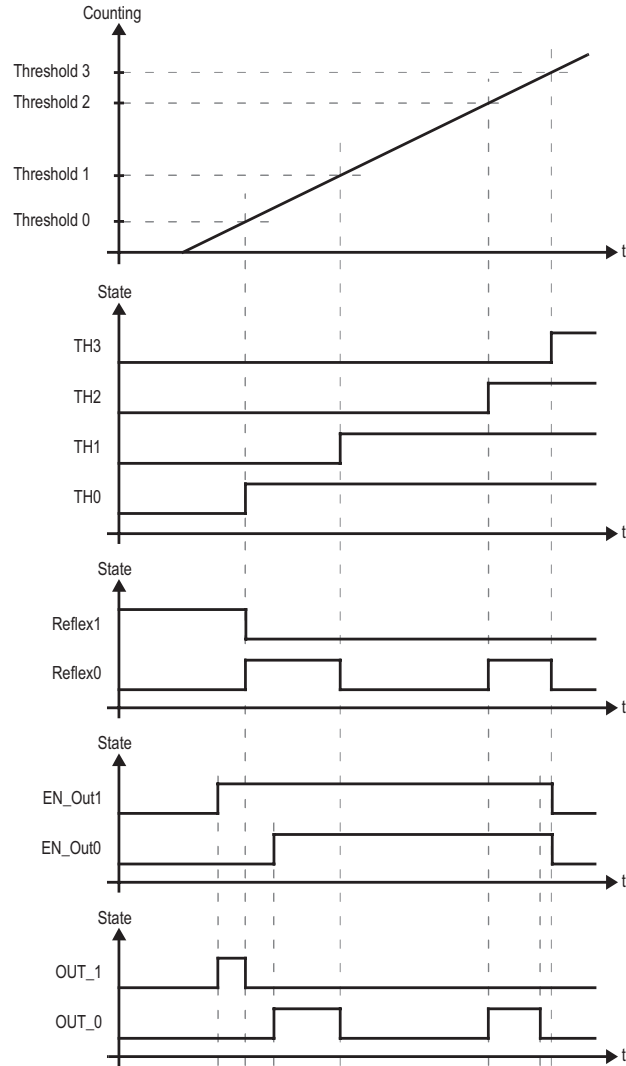
Reflex Output Behavior

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 or an Encoder 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, you must, at least, suspend 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 = 1`.
- Ensure 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:

Step	Action
1	<p>Set <code>SuspendCompare</code> to 1.</p> <p>The comparison is frozen at the current value:</p> <ul style="list-style-type: none"> • <code>TH0</code>, <code>TH1</code>, <code>Reflex0</code>, <code>Reflex1</code>, <code>Out0</code>, <code>Out1</code> output bits of the block maintain their last value. • Physical Outputs 0, 1 maintain their last value • Events are masked <p>NOTE: <code>EN_Compare</code>, <code>EN_Out0</code>, <code>EN_Out1</code>, <code>F_Out0</code>, <code>F_Out1</code> remain operational while <code>SuspendCompare</code> is set.</p>
2	<p>Modify the Threshold values as needed using the <code>EXPERTSetParam</code> (see page 196) function block.</p> <p>NOTE: Follow this rule to configure the threshold values: <code>TH0 < TH1 < TH2 < TH3</code>.</p>
3	<p>Set <code>SuspendCompare</code> to 0.</p> <p>The new Threshold values are applied and the comparison is resumed.</p>

Configuration of the Comparison on a Main Type or an Encoder

Configuration Window

Follow this procedure to configure the comparison function on a **Main** type or an Encoder:

Step	Action
1	Enter the Configuration Window
2	Double click the controller
3	Select: Expert I/O → DM72F0/DM72F1 → HSCMain or StandardEncoder/MotionEncoder
4	Enable the thresholds to use.
5	Provide the threshold value. NOTE: Follow this rule to configure the threshold values: TH0 < TH1 < TH2 < TH3
6	Optionally, provide an event condition (see page 159).
7	Optionally, configure the Reflex Output behavior. (see page 154)

External Event Configuration

Procedure

The following procedure describes how to configure an external event (see *Modicon M258 Logic Controller, Programming Guide*) to activate a task:

Step	Action
1	Add a task by left clicking on the task configuration node.
2	In the Program window, double click the task to associate to an external event.
3	In the Type drop menu, select External .
4	Select in the External Event drop menu the event to associate to the task. (see list below)

External Events

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

Event name	Description
BLOCK0_I0	Task is activated when the input I0 of the block DM72F0 is set to 1.
BLOCK0_I1	Task is activated when the input I1 of the block DM72F0 is set to 1.
BLOCK0_I2	Task is activated when the input I2 of the block DM72F0 is set to 1.
BLOCK0_I3	Task is activated when the input I3 of the block DM72F0 is set to 1.
BLOCK1_I4	Task is activated when the input I4 of the block DM72F1 is set to 1.
BLOCK1_I5	Task is activated when the input I5 of the block DM72F1 is set to 1.
BLOCK1_I6	Task is activated when the input I6 of the block DM72F1 is set to 1.
BLOCK1_I7	Task is activated when the input I7 of the block DM72F1 is set to 1.
BLOCK0_TH0	Task is activated when the threshold TH0 of the HSC or the encoder of the block DM72F0 is set to 1.
BLOCK0_TH1	Task is activated when the threshold TH1 of the HSC or the encoder of the block DM72F0 is set to 1.
BLOCK0_TH2	Task is activated when the threshold TH2 of the HSC or the encoder of the block DM72F0 is set to 1.
BLOCK0_TH3	Task is activated when the threshold TH3 of the HSC or the encoder of the block DM72F0 is set to 1.
BLOCK1_TH0	Task is activated when the threshold TH0 of the HSC or the encoder of the block DM72F1 is set to 1.
BLOCK1_TH1	Task is activated when the threshold TH1 of the HSC or the encoder of the block DM72F1 is set to 1.

Event name	Description
BLOCK1_TH2	Task is activated when the threshold TH2 of the HSC or the encoder of the block DM72F1 is set to 1.
BLOCK1_TH3	Task is activated when the threshold TH3 of the HSC or the encoder of the block DM72F1 is set to 1.
ENCODER_TH0	Task is activated when the threshold TH0 of an encoder of the Encoder interface is set to 1.
ENCODER_TH1	Task is activated when the threshold TH1 of an encoder of the Encoder interface is set to 1.
ENCODER_TH2	Task is activated when the threshold TH2 of an encoder of the Encoder interface is set to 1.
ENCODER_TH3	Task is activated when the threshold TH3 of an encoder of the Encoder interface is set to 1.
BLOCK0_STOPEVENT	Task is activated when the value of the HSC related to the block DM72F0 reaches 0 in One shot mode.
BLOCK1_STOPEVENT	Task is activated when the value of the HSC related to the block DM72F1 reaches 0 in One shot mode.

Capture Function

20

Overview

This chapter provides information on capture function for HSC.

What's in this Chapter?

This chapter contains the following sections:

Section	Topic	Page
20.1	Capture with a Main Type	162
20.2	Capture with an Encoder	165

20.1 Capture with a Main Type

Overview

This section provides information on the capture function with a **Main** type.

What's in this Section?

This section contains the following topics:

Topic	Page
Capture Principle with a Main type	163
Configuration of the Capture on a Main Type	164

Capture Principle with a Main type

Overview

The capture function stores the current counter value upon an external input signal.

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

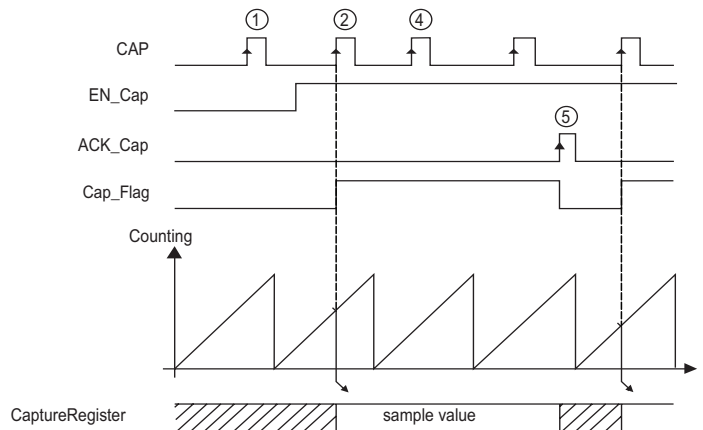
- One-shot (see page 45)
- Modulo-loop (see page 67)
- Free-large (see page 85)

Using this function requires to:

- configure the optional Capture input: **CAP**
- use `EXPERTGetCapturedValue` (see page 191) 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 <code>EN_Cap = 0</code> , the function is not operational.
2	When <code>EN_Cap = 1</code> , the edge on CAP captures the current counter value and puts it into the Capture register, and triggers the rising edge of <code>Cap_Flag</code> .
3	Get the stored value using <code>EXPERTGetCapturedValue</code> (see page 191).
4	While <code>Cap_Flag = 1</code> , any new edge on the physical input <code>CAP</code> is ignored.
5	The rising edge of <code>HSCMain_M258</code> (see page 203) function block input <code>ACK_Cap</code> triggers the falling edge <code>Cap_Flag</code> output. A new capture is authorized.

Configuration of the Capture on a Main Type

Configuration Window

The screenshot shows a configuration window with three sections: **Input Usage**, **Preset Condition**, and **Capture Mode**. In the **Input Usage** section, there are six rows: A, B, SYNC (I2), CAP (I3), EN (I4), and REF (I5). Each row has a checkbox and a dropdown menu. The CAP (I3) checkbox is checked, and its dropdown menu is open, showing '0.002'. The CAP mode dropdown is also open, showing 'Preset'. Numbered callouts 4, 5, and 6 point to the CAP input value, the CAP input dropdown, and the CAP mode dropdown respectively.

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

Step	Action
1	Enter the Configuration Window
2	Double click the controller
3	Select: Expert I/O → DM72F0/DM72F1 → HSCMain
4	Enable the CAP input.
5	Select a filtering value for CAP input.
6	Define the triggering edge of the CAP input.

20.2 Capture with an Encoder

Overview

This section provides informations on the capture function with an **Standard Encoder** or **Motion Encoder**.

What's in this Section?

This section contains the following topics:

Topic	Page
Capture with an Encoder	166
Configuration of the Capture on an Encoder	169

Capture with an Encoder

Overview

The capture function stores the current counter value upon an external input signal.

Each Encoder have 2 registers of capture (CAP0 and CAP1). Those registers can be used in 2 ways:

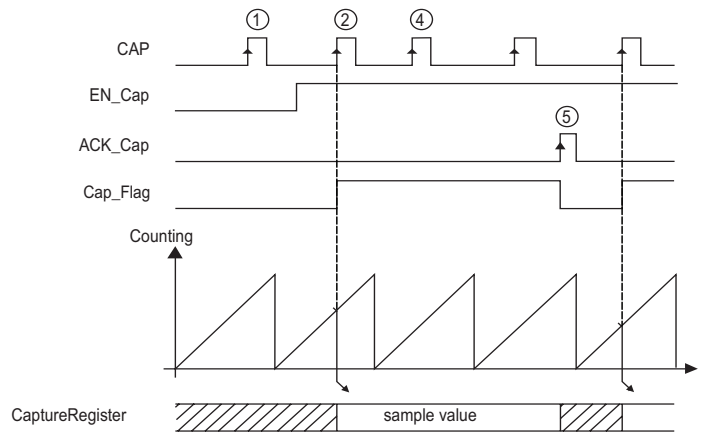
- Up to 2 position captures
- 1 distance capture

Using this function requires to:

- configure optional Capture inputs: **CAP**
- use `EXPERTGetCapturedValue` (see page 191) function block to retrieve the captured value in your application.

Principle of a Capture

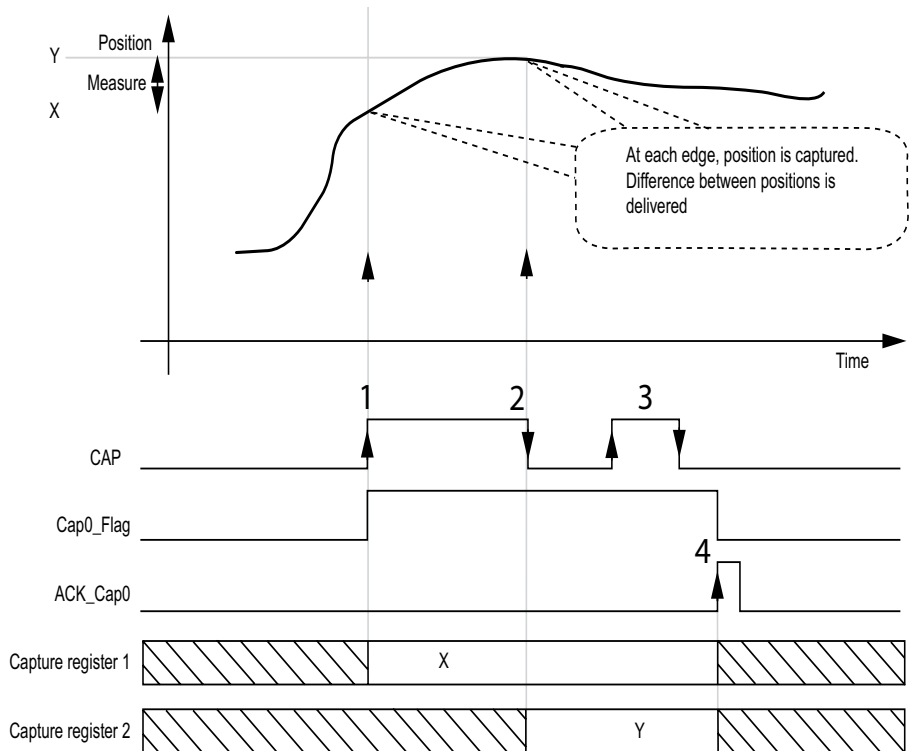
This graphic illustrates how the position capture works (only one register is shown):



Stage	Action
1	When <code>En_Cap = 0</code> , the function is not operational.
2	When <code>EN_Cap = 1</code> , the edge on CAP captures the current counter value and puts it into the Capture register, and triggers the rising edge of <code>Cap_Flag</code> .
3	Get the stored value using <code>EXPERTGetCapturedValue</code> (see page 191).
4	While <code>Cap_Flag = 1</code> , any new edge on the physical input CAP is ignored.
5	The rising edge of Encoder (see page 203) function block input <code>ACK_Cap</code> triggers the falling edge <code>Cap_Flag</code> output. A new capture is authorized.

Principle of Distance Capture

When using an encoder, the distance capture allows to get the difference between each edge of **CAP** input as shown in the following diagram:



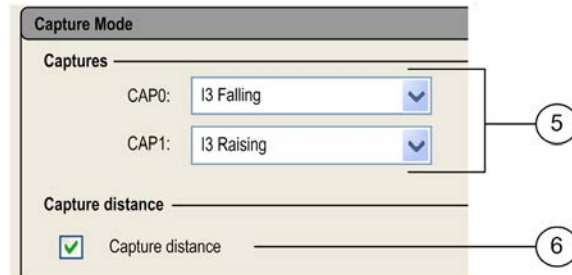
Stage	Action
1	The rising edge of CAP captures the current counter value and puts it into the first capture register.
2	The falling edge of CAP captures the current counter value and puts it into the second Capture register, and triggers the rising edge of Cap0_Flag.
3	Get the stored value using EXPERTGetCapturedValue (see page 191). The EXPERTGetCapturedValue (see page 191) function block can get: <ul style="list-style-type: none"> the position at the rising edge the position at the falling edge distance value
4	While Cap0_Flag = 1, any new edge on the physical input CAP is ignored.

Stage	Action
5	The rising edge of <code>Encoder</code> (see page 203) function block input <code>ACK_Cap</code> triggers the falling edge <code>Cap_Flag</code> output. A new capture is authorized.

NOTE: In the case of a rotary axis, the distance is always positive even if the position at the falling edge is smaller than the position at the rising edge.

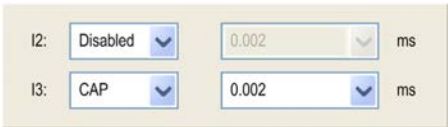
Configuration of the Capture on an Encoder

Configuration Window on a Expert I/O Module



Configuration Procedure on a Expert I/O Module

Follow this procedure to configure the capture function of an **Encoder**:

Step	Action
1	Enter the Configuration Window
2	Double click the controller
3	Select Expert I/O → DM72F0/DM72F1 → Standard Encoder
4	Enable the CAP and/or Z inputs and provide a filtering value: 
5	Define the triggering condition of CAP0 and/or the CAP1. NOTE: You cannot choose the same condition for CAP0 and CAP1 .
6	Tick the Capture distance check box to enable the capture distance function

Preset and Enable Functions

21

Overview

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

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Preset Function	172
Free-large or Period Meter Preset Conditions	174
Enable Function	176

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:

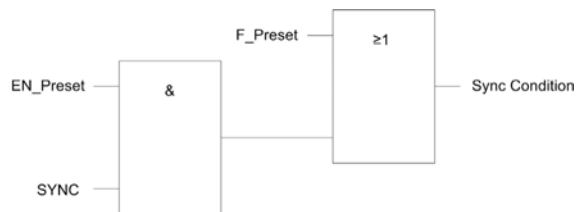
- **One shot** counter: preset and start the counter
- **Modulo-loop** counter: reset and start the counter
- **Event counting**: restart the internal time base at the beginning

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

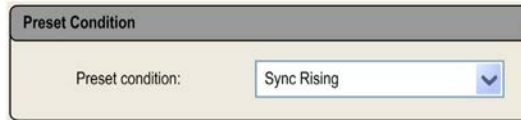
SYNC physical input `SYNC`

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

The `Sync condition` operates on a rising edge.

Configuration

The transition type of the SYNC physical input is determined by the **Preset Condition** parameter.



The image shows a software configuration window titled "Preset Condition". Inside the window, there is a label "Preset condition:" followed by a dropdown menu. The dropdown menu is currently set to "Sync Rising".

There are 3 available transition, defined by configuration:

- Rising edge of the SYNC input
- Falling edge of the SYNC input
- Both edge of the SYNC input

Free-large or Period Meter Preset Conditions

Overview

In **Free-large** mode, the Preset condition is create by using 2 inputs:

- SYNC (respectively Z for the Encoder)
- REF

There are 7 preset conditions available:

- At the edge of the input SYNC (rising, falling or both)
- At the rising edge of the input REF
- At the rising edge of the input SYNC if the input REF is active high
- At the first SYNC pulse after the REF input signal rising
- At the first SYNC pulse after the REF input signal falling

At the Edge of the Input SYNC (Rising, Falling or Both)

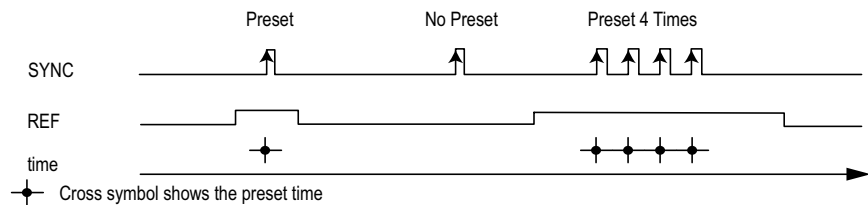
The counter synchronizes upon the encoder reference point.

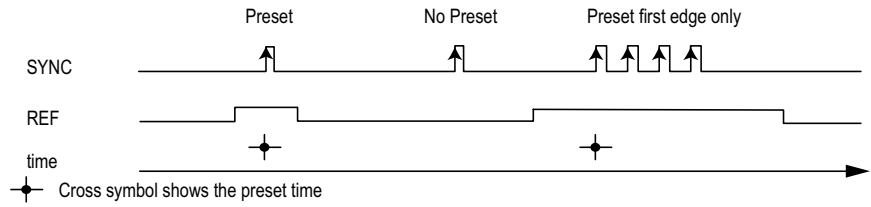
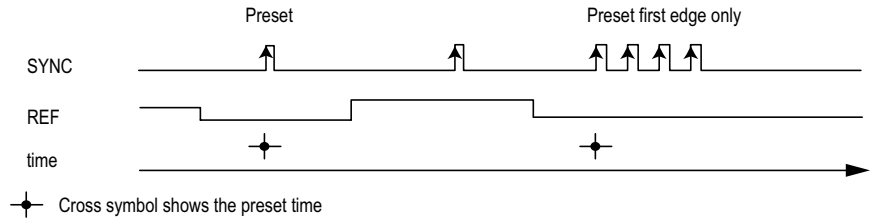
At the Rising Edge of the Input REF

The counter synchronizes upon the mechanical position.

At the Rising Edge of the Input SYNC if the Input REF is Active High

The counter synchronizes upon the encoder reference point when the **REF** signal is **TRUE**, as shown below:.



At the First SYNC Pulse after the REF Input Signal Rising:**At the first SYNC Pulse after the REF Input Signal Falling:**

Enable Function

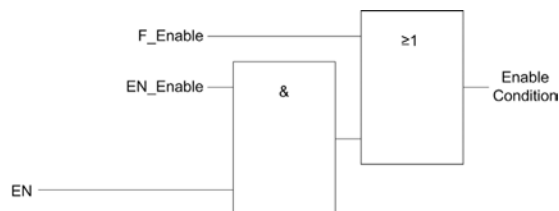
Overview

The enable function is used to authorize the counting operation.

Description

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

The following diagrams 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 inputs **Enable**.

Appendices



Overview

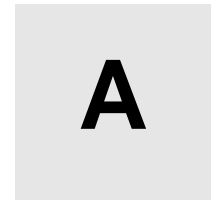
This appendix extracts parts of the programming guide for technical understanding of the library documentation.

What's in this Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	General Information	179
B	Data Types	183
C	Function Blocks	189
D	Function and Function Block Representation	211

General Information



Overview

The information described in this chapter is common for PTO and HSC administrative and motion functions.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
Dedicated Functions	180
General Information on Administrative and Motion Function Block Management	181

Dedicated Functions

Dedicated Outputs

Outputs used by the Pulse Train Output, Frequency Generator, Pulse Width Modulation, High Speed Counters or an Encoder can only be accessed through the function block. They can not be read or written directly within the application.

When using these dedicated functions, observe the following precautions to avoid unintended equipment operation with the functions and the equipment they control:

- Do not use the same function block instance in different program tasks.
- Do not change the function block reference (****_REF_IN**) while the function block is active (executing).

WARNING

UNINTENDED EQUIPMENT OPERATION

- Do not use the same instance of a function block in more than 1 task.
- Do not modify function block references (****_REF_IN**) while the function block is active (executing).

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

General Information on Administrative and Motion Function Block Management

Management of Input Variables

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

Any further modifications of the input variables are not taken in 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 have known values attributed to its inputs to help avoid difficulties in debugging your program. For HSC and PTO function blocks, it is best to use the instance only once, and that instance must be in the main task.

Management of Output Variables

The `Done`, `InVelocity`, or `InFrequency` output is mutually exclusive with `Busy`, `CommandAborted`, 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`, `InVelocity`, `InFrequency`, `CommandAborted`, and `Error`).

The `Done`, `InVelocity`, or `InFrequency` output is set when the function block execution has been completed successfully.

When a function block execution is interrupted by another one, the `CommandAborted` output is set instead.

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

The `Done`, `InVelocity`, `InFrequency`, `Error`, `ErrID`, and `CommandAborted` outputs are reset with the falling edge of `Execute`. If `Execute` input is reset before the 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, like `Done`, for the previous action.

Error Handling

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

- `Error` = `TRUE` when an error is detected.
- `ErrID` When `Error` = `TRUE`, returns the detected error ID.

Data Types



B

Overview

This chapter describes the data types of the HSC Library.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
IMMEDIATE_ERR_TYPE: Type for Error Variable of the GetImmediateValue Function Block	184
EXPERT_ERR_TYPE: Type for Error Variable of EXPERT Function Block	185
EXPERT_PARAMETER_TYPE: Type for Parameters to Get or to Set on EXPERTFunction Block	186
EXPERT_REF: EXPERT Reference Value	187
EXPERT_TIMEBASE_TYPE: Type for HSC Time Base Variable	188

IMMEDIATE_ERR_TYPE: Type for Error Variable of the GetImmediateValue Function Block**Enumerated Type Description**

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

Enumerator	Value	Description
IMMEDIATE_NO_ERROR	0	No error detected
IMMEDIATE_UNKNOWN	1	The reference of IMMEDIATE function is incorrect or not configured
IMMEDIATE_UNKNOWN_PARAMETER	2	A parameter reference is incorrect

EXPERT_ERR_TYPE: Type for Error Variable of EXPERT Function Block

Enumerated Type Description

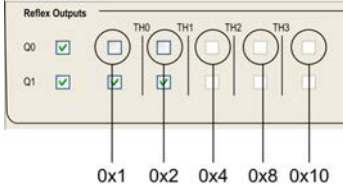
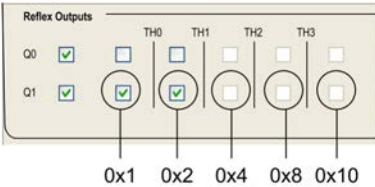
The enumeration data type ENUM contains the different types of detected error with the following values:

Enumerator	Value	Description
EXPERT_NO_ERROR	00 hex	No error detected
EXPERT_UNKNOWN	01 hex	The reference EXPERT is incorrect or not configured.
HSC_UNKNOWN_PARAMETER	02 hex	The parameter reference is incorrect.
EXPERT_INVALID_PARAMETER	03 hex	The value of the parameter is incorrect.
EXPERT_COM_ERROR	04 hex	Communication error was detected with the EXPERT module.
EXPERT_CAPTURE_NOT_CONFIGURED	05 hex	Capture is not configured.

EXPERT_PARAMETER_TYPE: Type for Parameters to Get or to Set on EXPERTFunction Block

Enumerated Type Description

The enumeration data type ENUM contains the following values:

Enumerator	Value	Description
EXPERT_PRESET	00 hex	To get or set the Preset (Offset for an Encoder) value of an EXPERT function.
EXPERT_MODULO	01 hex	To get or set the Modulo value of an EXPERT function.
EXPERT_OFFSET	02 hex	to get or set OFFSET of an EXPERT function
EXPERT_TIMEBASE	03 hex	To get or set the Timebase value (<i>see page 188</i>) of an EXPERT function.
EXPERT_SLACK	04 hex	To get or set the Slack value of an EXPERT function (only for Encoder).
EXPERT_SCALING	05 hex	to get or set the scaling parameter. The scaling parameter (ParamValue of FB) is made of 2 sub-parameters: High significant INT = Increments Low significant INT = Units
EXPERT_THRESHOLD0	06 hex	To get or set the Threshold 0 value of an EXPERT function.
EXPERT_THRESHOLD1	07 hex	To get or set the Threshold 1 value of an EXPERT function.
EXPERT_THRESHOLD2	08 hex	To get or set the Threshold 2 value of an EXPERT function.
EXPERT_THRESHOLD3	09 hex	To get or set the Threshold 3 value of an EXPERT function.
EXPERT_REFLEX0	0A hex	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: 
EXPERT_REFLEX1	0B hex	to get or set output 0 reflex mode of an EXPERT function The five less significant bits correspond to a reflex mode: 

EXPERT_REF: EXPERT Reference Value**Data Type Description**

The EXPERT_REF is a byte used to identify the EXPERT function associated with the administrative block.

EXPERT_TIMEBASE_TYPE: Type for HSC Time Base Variable**Enumerated Type Description**

The enumeration data type ENUM contains the different timebase values allowed for use with an EXPERT function block:

Name	Value
EXPERT_100ms	00 hex
EXPERT_1s	01 hex
EXPERT_10s	02 hex
EXPERT_60s	03 hex

Function Blocks



C

Overview

This chapter describes the functions and the function blocks of the HSC and encoder part of the EXPERT I/O Library.

What's in this Chapter?

This chapter contains the following topics:

Topic	Page
EXPERTGetImmediateValue: Read Counter Value of HSC or Encoder Function	190
EXPERTGetCapturedValue: Returns Content of Capture Registers	191
EXPERTGetDiag: Provides Detail of Detected Error on a Principal EXPERT I/O Function	193
EXPERTGetParam: Returns Parameters of Principal EXPERT I/O Function	196
EXPERTSetParam: Adjust Parameters of a HSC	198
Encoder_M258: Encoder Function Block	200
HSCMain_M258: HSC Main Function Block	203
HSCSimple_M258: HSC Simple Function Block	208

EXPERTGetImmediateValue: Read Counter Value of HSC or Encoder Function

Function Description

This administrative function permits to read the counter value of an HSC or Encoder bypassing the controller cycle.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the *Function and Function Block Representation* (see page 211) chapter.

I/O Variables Description

The following table describes the input variables:

Inputs	Type	Comment
EXPERT_REF	EXPERT_REF (see page 187)	Reference to the EXPERT function block.

The following table describes the output variables:

Outputs	Type	Comment
EXPERTGetImmediateValue	DINT	Contains the counter value.

The following table describes the input/output variables:

Input/Output	Type	Comment
Error	BOOL	TRUE = indicates that an error was detected.
ErrID	IMMEDIATE_FUNC_ERR_TYPE (see page 184)	When Error is TRUE: type of the detected error.

EXPERTGetCapturedValue: Returns Content of Capture Registers

Function Description

This administrative function block returns the content of a capture register.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the *Function and Function Block Representation* (see page 211) chapter.

I/O Variables Description

The following table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF (see page 187)	Reference to the EXPERT 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.
CaptureNumber	BYTE	Index of the capture register: <ul style="list-style-type: none"> • for HSCMain: always 0 • for Encoder: 0: Cap0, 1: Cap1, or 2: Distance

The following table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF (see page 187)	Reference to the EXPERT function block.
Done	BOOL	TRUE = indicates that CaptureValue 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.
ErrID	EXPERT_ERR_TYPE (see page 185)	When Error is TRUE: type of the detected error
CaptureValue	DINT	When Done is TRUE: Capture register value is valid.

NOTE: In case of detected error, variables take the last value captured.

NOTE: For more information about Done, Busy and Execution pins, refer to General Information on Function Block Management (see page 181).

Adding the EXPERTGetCapturedValue Function Block

Step	Description
1	Insert the EXPERTGetCapturedValue function block using the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP → Administrative
2	Link the EXPERT_REF_IN input to the HSC_REF output of the HSC.

EXPERTGetDiag: Provides Detail of Detected Error on a Principal EXPERT I/O Function

Function Description

This administrative function block returns the details of a detected HSC error.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the *Function and Function Block Representation* (see page 211) chapter.

I/O Variables Description

The following table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF (see page 187)	Reference to the EXPERT 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.

The following table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF (see page 187)	Reference to the EXPERT function block.
Done	BOOL	TRUE = indicates that HSCdiag is valid. Function block execution is finished.
Busy	BOOL	TRUE = indicates that the function block execution is in progress.

Outputs	Type	Comment
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrID	EXPERT_ERR_TYPE (see page 185)	When Error is TRUE: type of the detected error
EXPERTDiag	DWORD	When Done is TRUE: diagnostic value is valid, refer to the table below.

NOTE: For more information about Done, Busy and Execution pins, refer to General Information on Function Block Management (see page 181).

This table indicates the diagnostic values:

Bit	HSC	Standard Encoder
0	Error detected on physical inputs	
1	–	Error detected on physical outputs
2	–	–
3	–	–
4	–	Encoder power distribution feedback
5 ⁽¹⁾	–	Error detected on the transmission of the absolute SSI encoder frame
6 ⁽¹⁾	–	Indicates a parity error detected of the absolute SSI encoder frame
7	Invalid Configuration Detected	
8	Invalid adjustment parameters detected	
9	–	Encoder configuration in progress
10	–	–
11 ⁽¹⁾	–	Absolute SSI encoder status bit 0. Refer to your encoder user guide.
12 ⁽¹⁾	–	Absolute SSI encoder status bit 1. Refer to your encoder user guide.
13 ⁽¹⁾	–	Absolute SSI encoder status bit 2. Refer to your encoder user guide.
14 ⁽¹⁾	–	Absolute SSI encoder status bit 3. Refer to your encoder user guide.
15 ⁽¹⁾	–	–
(1) In case of a detected SSI error set the Enable condition (see page 176) to 0 to acknowledge the error condition.		

Adding the EXPERTGetDiag Function Block

Step	Description
1	Insert the <code>EXPERTGetDiag</code> function block using the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP → Administrative
2	Link the <code>EXPERT_REF_IN</code> input to the <code>HSC_REF</code> output of the HSC.

EXPERTGetParam: Returns Parameters of Principal EXPERT I/O Function

Function 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 the *Function and Function Block Representation (see page 211)* chapter.

I/O Variables Description

The following table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF (see page 187)	Reference to the EXPERT 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	EXPERT_PARAMETER_TYPE (see page 186)	Parameter to read.

The following table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF (see page 187)	Reference to the EXPERT function block.
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.
ErrID	EXPERT_ERR_TYPE (see page 185)	When Error is TRUE: type of the detected error
ParamValue	DINT	Value of the parameter that has been read.

NOTE: For more information about Done, Busy and Execution pins, refer to General Information on Function Block Management (see page 181).

Adding the EXPERTGetParam Function Block

Step	Description
1	Insert the EXPERTGetParam function block using the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP → Administrative
2	Link the EXPERT_REF_IN input to the HSC_REF output of the HSC.

EXPERTSetParam: Adjust Parameters of a HSC

Function 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 the *Function and Function Block Representation* (see page 211) chapter.

I/O Variables Description

The following table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF (see page 187)	Reference to the EXPERT 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	EXPERT_PARAMETER_TYPE (see page 186)	Parameter to read.
ParamValue	DINT	Parameter value to write.

The following table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF (see page 187)	Reference to the EXPERT function block.
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.
ErrID	EXPERT_ERR_TYPE (see page 185)	When Error is TRUE: type of the detected error

NOTE: For more information about Done, Busy and Execution pins, refer to General Information on Function Block Management (see page 181).

Adding the EXPERTSetParam Function Block

Step	Description
1	Insert the EXPERTSetParam function block using the insert box assistant. The function block can be found at this path: Function Block (Libraries) → SEC_EXP → Administrative
2	Link the EXPERT_REF_IN input to the HSC_REF output of the HSC.

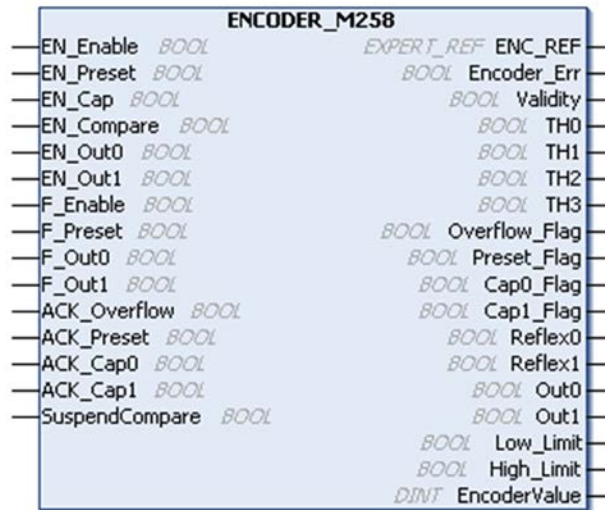
Encoder_M258: Encoder Function Block

Function Description

This function block controls a Encoder type counter.

The function block instance name must match the name defined by the configuration.

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the *Function and Function Block Representation* (see page 211) chapter.

I/O Variables Description

The following table describes the input variables:

Inputs	Type	Comment
EN_Enable	BOOL	TRUE = authorizes the encoder enable via the Enable input (if configured).
EN_Preset	BOOL	TRUE = authorizes the encoder synchronization and starts via the Sync input (if configured).
EN_Cap	BOOL	TRUE = enables the Capture input (if configured).

Inputs	Type	Comment
EN_Compare	BOOL	TRUE = enables the comparator operation (using Thresholds 0, 1, 2, 3): <ul style="list-style-type: none"> ● basic comparison (TH0, TH1, TH2, TH3 output bits) ● reflex (Reflex0, Reflex1 output bits) ● events (to trigger external tasks on threshold crossing)
EN_Out0	BOOL	TRUE = enables Output0 to echo the Reflex0 value (if configured, on DM72F modules).
EN_Out1	BOOL	TRUE = enables Output1 to echo the Reflex1 value (if configured, on DM72F modules).
F_Enable	BOOL	Forces the Enable condition (<i>see page 176</i>).
F_Preset	BOOL	Forces the Preset condition.
F_Out0	BOOL	TRUE = forces Output0 to 1 (if Reflex0 is configured).
F_Out1	BOOL	TRUE = forces Output1 to 1 (if Reflex1 is configured).
ACK_Overflow	BOOL	On rising edge, resets the Overflow_Flag.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap0	BOOL	On rising edge, resets Cap0_Flag.
ACK_Cap1	BOOL	On rising edge, resets Cap1_Flag.
SuspendCompare	BOOL	TRUE = the comparator operation results are frozen: <ul style="list-style-type: none"> ● TH0, TH1, TH2, TH3, Reflex0, Reflex1 output bits maintain their last value. ● Hardware Outputs 0, 1 maintain their last value. ● Events are masked. EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.

The following table describes the output variables:

Outputs	Type	Comment
ENC_REF	EXPERT_REF (<i>see page 187</i>)	Reference to the Encoder.
Encoder_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (<i>see page 193</i>) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that the output values on the function block are valid.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 (if configured). Only active when EN_Compare is set.

Outputs	Type	Comment
TH1	BOOL	Set to 1 when <code>CurrentValue > Threshold 1</code> (if configured). Only active when <code>EN_Compare</code> is set.
TH2	BOOL	Set to 1 when <code>CurrentValue > Threshold 2</code> (if configured). Only active when <code>EN_Compare</code> is set.
TH3	BOOL	Set to 1 when <code>CurrentValue > Threshold 3</code> (if configured). Only active when <code>EN_Compare</code> is set.
Overflow_Flag	BOOL	Set to 1 when the encoder rolls over its limits.
Preset_Flag	BOOL	Set to 1 after the encoder presets (see page 174).
Cap0_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register. This flag must be reset before a new capture is allowed.
Cap1_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register. This flag must be reset before a new capture is allowed.
Reflex0	BOOL	State of <code>Reflex0</code> (if configured). Only active when <code>EN_Compare</code> is set.
Reflex1	BOOL	State of <code>Reflex1</code> (if configured). Only active when <code>EN_Compare</code> is set.
Out0	BOOL	Indicates the state of Output0.
Out1	BOOL	Indicates the state of Output1.
Low_Limit	BOOL	Only managed for incremental linear in Lock on limit. Set to 1 when the encoder exceeds - 2.147.483.648. Resets to 0 when encoder presets or resets.
High_Limit	BOOL	Only managed for incremental linear in Lock on limit. Set to 1 when the encoder exceeds + 2.147.483.648. Resets to 0 when encoder presets or resets.
EncoderValue	DINT	Current value of the Encoder.

HSCMain_M258: HSC Main Function Block

Function Description

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

- up/down counting
- frequency-meter
- thresholds
- events

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

The function block instance name must match the name defined by the configuration.

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 SoMachine 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 the *Function and Function Block Representation* (see page 211) chapter.

I/O Variables Description

The following table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	TRUE = enables the counter.
EN_Preset	BOOL	TRUE = authorizes the counter synchronization and start via 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): <ul style="list-style-type: none"> ● basic comparison (TH0, TH1, TH2, TH3 output bits) ● reflex (Reflex0, Reflex1 output bits) ● events (to trigger external tasks on threshold crossing)
EN_Out0	BOOL	TRUE = enables Output0 to echo the Reflex0 value (if configured in One shot, Modulo loop, Free large modes).

Input	Type	Description
EN_Out1	BOOL	TRUE = enables Output1 to echo the Reflex1 value (if configured in One shot, Modulo loop, Free large modes).
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
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 time base at the beginning
F_Out0	BOOL	TRUE = forces Output0 to 1 (if configured in One shot, Modulo loop, Free large modes).
F_Out1	BOOL	TRUE = forces Output1 to 1 (if configured in One shot, Modulo loop, Free large modes).
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag (Modulo loop and Free large modes).
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets the Cap_Flag (One shot, Modulo loop, Free large modes).
SuspendCompare	BOOL	TRUE = compare results are suspended: <ul style="list-style-type: none"> ● TH0, TH1, TH2, TH3, Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value. ● Physical outputs Output0 and Output1 maintain their last value. ● Events are masked. NOTE: EN_Compare, EN_Out0, EN_Out1, F_Out0, F_Out1 remain operational while SuspendCompare is set

The following table describes the output variables:

Outputs	Type	Comment
HSC_REF (see page 187)	EXPERT_REF (see page 187)	Reference to the HSC.
Error	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (see page 193) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid. In the Period Meter Mode , if the time out value is exceeded, Validity = FALSE.
Run	BOOL	TRUE = counter is running. In One shot mode , the Run bit switches to 0 when CurrentValue reaches 0.
TH0	BOOL	TRUE = current counter value > Threshold 0 (if configured in One shot, Modulo loop, Free large modes). Only active when EN_Compare is set.
TH1	BOOL	TRUE = current counter value > Threshold 1 (if configured in One shot, Modulo loop, Free large modes). Only active when EN_Compare is set.
TH2	BOOL	TRUE = current counter value > Threshold 2 (if configured in One shot, Modulo loop, Free large modes). Only active when EN_Compare is set.
TH3	BOOL	TRUE = current counter value > Threshold 3 (if configured in One shot, Modulo loop, Free large modes). Only active when EN_Compare is set.
Modulo_Flag	BOOL	Set to 1 by the rolls over of: <ul style="list-style-type: none"> ● Modulo loop counter: when the counter rolls over to the modulo or 0 ● Free large counter: when the counter roll overs its limits

Outputs	Type	Comment
Preset_Flag	BOOL	Set to 1 by the synchronization of: <ul style="list-style-type: none"> ● 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.
Reflex0	BOOL	State of Reflex0 (if configured in One shot, Modulo loop, Free large modes). Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1 (if configured in One shot, Modulo loop, Free large modes). Only active when EN_Compare is set.
Out0	BOOL	Indicates the state of Output0
Out1	BOOL	Indicates the state of Output1
CurrentValue	DINT	Current counter value of the counter.

HSCSimple_M258: HSC Simple Function Block

Function Description

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

- one-way counting
- no threshold

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

The function block instance name must match the name defined by the configuration.

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 SoMachine 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 the *Function and Function Block Representation (see page 211)* chapter.

I/O Variables Description

The following table describes the input variables:

Inputs	Type	Comment
Enable	BOOL	TRUE = authorizes changes to the current counter value.
Sync	BOOL	On rising edge, presets and starts the counter
ACK_Modulo	BOOL	On rising edge, resets the modulo flag Modulo_Flag (in Modulo loop mode).

The following table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF (see page 187)	Reference to the HSC.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (see page 193) function block used to get more information about this detected error.
Validity	BOOL	TRUE = indicates that the output values on the function block are valid.
Run	BOOL	TRUE = counter is running. In One-shot mode, Switches to 0 when CurrentValue reaches 0. A rising edge on Sync is needed to restart the counter.
Modulo_Flag	BOOL	Set to 1 by the rolls over of a Modulo loop counter: when the counter rolls over the modulo.
CurrentValue	DWORD	Current count value of the counter.

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's in this Chapter?

This chapter contains the following topics:

Topic	Page
Differences Between a Function and a Function Block	212
How to Use a Function or a Function Block in IL Language	213
How to Use a Function or a Function Block in ST Language	216

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
- is always called through 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

Examples: timers, counters

In the example below, `Timer_ON` is an instance of the Function Block `TON`:

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

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

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

General Information

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

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

Using a Function in IL Language

The following 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 the SoMachine global help.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: <ul style="list-style-type: none"> • type the name of the function in the operator column (left field), or • use the Input Assistant to select the function (select Insert Box in context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with ??? in the fields on the right. Replace the ??? with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

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

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

In IL language, the function name is used directly in the **Operator Column**:

Function	Representation in SoMachine POU IL Editor
IL example of a function without input parameter: IsFirstMastCycle	<pre> 1 PROGRAM MyProgram_IL 2 VAR 3 FirstCycle: BOOL; 4 END_VAR 5 </pre> <hr/> <pre> 1 IsFirstMastCycle ST FirstCycle </pre>
IL example of a function with input parameters: SetRTCDrift	<pre> 1 PROGRAM MyProgram_IL 2 VAR 3 myDrift: SINT (-29..29) := 5; 4 myDay: DAY_OF_WEEK := SUNDAY; 5 myHour: HOUR := 12; 6 myMinute: MINUTE; 7 myDiag: RTCSETDRIFT_ERROR; 8 END_VAR 9 </pre> <hr/> <pre> 1 LD myDrift SetRTCDrift myDay myHour myMinute ST myDiag </pre>

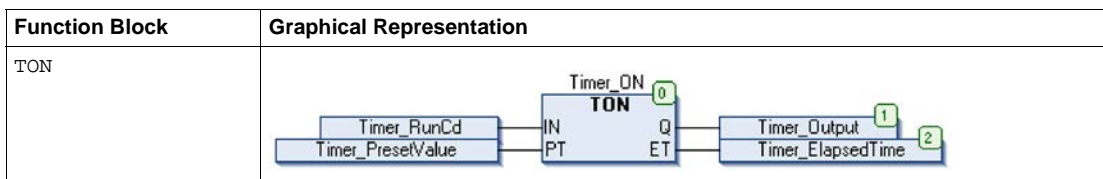
Using a Function Block in IL language

The following 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 the SoMachine global help.
2	Create the variables that the function block requires, including the instance name.

Step	Action
3	Function Blocks are called using a CAL instruction: <ul style="list-style-type: none"> ● Use the Input Assistant to select the FB (right-click and select Insert Box in context menu). ● Automatically, the CAL instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: <ul style="list-style-type: none"> ● Value to inputs are set by " := ". ● Values to outputs are set by " => ".
4	In the CAL right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

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



In IL language, the function block name is used directly in the **Operator Column**:

Function Block	Representation in SoMachine POU IL Editor
TON	<pre> 1 PROGRAM MyProgram_IL 2 VAR 3 Timer_ON: TON; // Function Block instance declaration 4 Timer_RunCd: BOOL; 5 Timer_PresetValue: TIME := T#5S; 6 Timer_Output: BOOL; 7 Timer_ElapsedTime: TIME; 8 END_VAR 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440 441 442 443 444 445 446 447 448 449 450 451 452 453 454 455 456 457 458 459 460 461 462 463 464 465 466 467 468 469 470 471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491 492 493 494 495 496 497 498 499 500 501 502 503 504 505 506 507 508 509 510 511 512 513 514 515 516 517 518 519 520 521 522 523 524 525 526 527 528 529 530 531 532 533 534 535 536 537 538 539 540 541 542 543 544 545 546 547 548 549 550 551 552 553 554 555 556 557 558 559 560 561 562 563 564 565 566 567 568 569 570 571 572 573 574 575 576 577 578 579 580 581 582 583 584 585 586 587 588 589 590 591 592 593 594 595 596 597 598 599 600 601 602 603 604 605 606 607 608 609 610 611 612 613 614 615 616 617 618 619 620 621 622 623 624 625 626 627 628 629 630 631 632 633 634 635 636 637 638 639 640 641 642 643 644 645 646 647 648 649 650 651 652 653 654 655 656 657 658 659 660 661 662 663 664 665 666 667 668 669 670 671 672 673 674 675 676 677 678 679 680 681 682 683 684 685 686 687 688 689 690 691 692 693 694 695 696 697 698 699 700 701 702 703 704 705 706 707 708 709 710 711 712 713 714 715 716 717 718 719 720 721 722 723 724 725 726 727 728 729 730 731 732 733 734 735 736 737 738 739 740 741 742 743 744 745 746 747 748 749 750 751 752 753 754 755 756 757 758 759 760 761 762 763 764 765 766 767 768 769 770 771 772 773 774 775 776 777 778 779 780 781 782 783 784 785 786 787 788 789 790 791 792 793 794 795 796 797 798 799 800 801 802 803 804 805 806 807 808 809 810 811 812 813 814 815 816 817 818 819 820 821 822 823 824 825 826 827 828 829 830 831 832 833 834 835 836 837 838 839 840 841 842 843 844 845 846 847 848 849 850 851 852 853 854 855 856 857 858 859 860 861 862 863 864 865 866 867 868 869 870 871 872 873 874 875 876 877 878 879 880 881 882 883 884 885 886 887 888 889 890 891 892 893 894 895 896 897 898 899 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 920 921 922 923 924 925 926 927 928 929 930 931 932 933 934 935 936 937 938 939 940 941 942 943 944 945 946 947 948 949 950 951 952 953 954 955 956 957 958 959 960 961 962 963 964 965 966 967 968 969 970 971 972 973 974 975 976 977 978 979 980 981 982 983 984 985 986 987 988 989 990 991 992 993 994 995 996 997 998 999 1000 </pre>

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

General Information

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

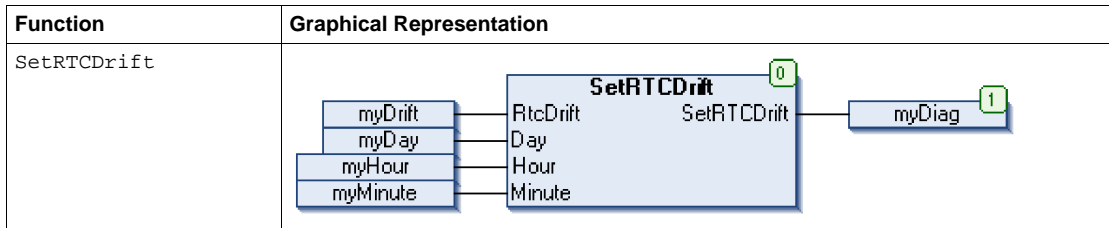
Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

The following 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 the SoMachine global help.
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: <code>FunctionResult := FunctionName(VarInput1, VarInput2,.. VarInputx);</code>

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



The ST language of this function is the following:

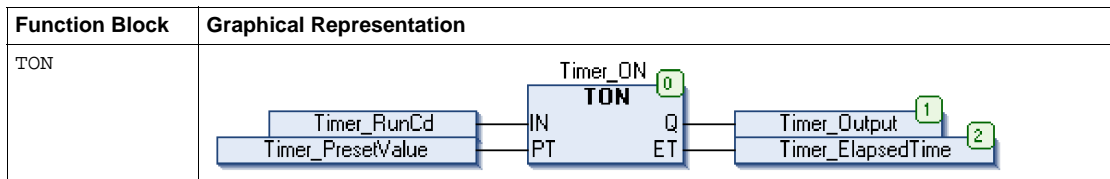
Function	Representation in SoMachine POU ST Editor
SetRTCDrift	<pre> PROGRAM MyProgram_ST VAR myDrift: SINT(-29..29) := 5; myDay: DAY_OF_WEEK := SUNDAY; myHour: HOUR := 12; myMinute: MINUTE; myRTCAadjust: RTCDRIFT_ERROR; END_VAR myRTCAadjust := SetRTCDrift(myDrift, myDay, myHour, myMinute); </pre>

Using a Function Block in ST Language

The following 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, refer to the SoMachine global help.
2	Create the input and output variables and the instance required for the function block: <ul style="list-style-type: none"> • Input variables are the input parameters required by the function block • Output variables receive the value returned by the function block
3	Use the general syntax in the POU ST Editor for the ST language of a Function Block. The general syntax is: FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2, ... Ouput1=>VarOutput1, Ouput2=>VarOutput2, ...);

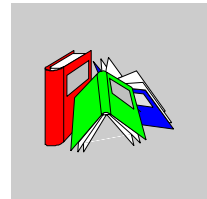
To illustrate the procedure, consider this example with the TON function block graphically presented below:



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

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

Glossary



0-9

%I

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

%IW

According to the IEC standard, %IW represents an input word register (for example a language object of type analog IN).

%MW

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

%Q

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

%QW

According to the IEC standard, %QW represents an output word register (for example a language object of type analog OUT).

1-phase counter

A *1-phase counter* uses 1 hardware input as counter input. It usually counts up or counts down when there is pulse signal in the input.

2-phase counter

A *2-phase counter* uses the phase difference between 2 input counter signals to count up or count down.

A

ADC

analog/digital converter

AFB

application function block

AMOA

An *address of modbus of option application board* installed on the drive.

analog input

An *analog input* module contains circuits that convert an analog DC input signal to a digital value that can be manipulated by the processor. By implication, the analog input is usually direct. That means a data table value directly reflects the analog signal value.

analog output

An *analog output* module contains circuits that transmit an analog DC signal proportional to a digital value input to the module from the processor. By implication, these analog outputs are usually direct. That means a data table value directly controls the analog signal value.

application source

The *application source* file can be uploaded to the PC to reopen a SoMachine project. This source file can support a full SoMachine project (for example, one that includes HMI application).

ARP

The *address resolution protocol* is the IP network layer protocol for Ethernet that maps an IP address to a MAC (hardware) address.

ARRAY

An **ARRAY** is a table containing elements of a single type. The syntax is as follows:
ARRAY [<limits>] OF <Type>

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

Example 2: ARRAY [1..10, 1..20] OF **INT** is a 2-dimensional table with 10x20 elements of type **INT**.

ARW

anti-reset windup

ASCII

The *american standard code for information interchange* is a communication protocol for representing alphanumeric characters (letters, numbers, and certain graphic and control characters).

assigned variable

A variable is "assigned" if its location in controller memory can be known. For example, the `Water_pressure` variable is said to be assigned through its association with memory location `%MW102.Water_pressure`.

ATC

analog tension control

ATV

ATV is the model prefix for Altivar drives. (For example, "ATV312" refers to the Altivar 312 variable speed drive.)

AWG

The *american wire gauge* standard specifies wire gauges in North America.

B**BCD**

The *binary coded decimal format* represents decimal numbers between 0 and 9 with a set of 4 bits (a nybble/nibble, also titled as Halfbyte). In this format, the 4 bits used to encode decimal numbers have an unused range of combinations. For example, the number 2,450 is encoded as 0010 0100 0101 0000

BOOL

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

Boot application

Files that contain machine dependent parameters:

- machine name
- device name or IP address
- Modbus Serial Line address
- Routing table

BOOTP

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

bps

bit per second as a definition of transmission rate, also given in conjunction with multiplier kilo (kbps) and mega (mbps).

BSH

BSH is a Lexium servo motor from Schneider Electric.

bus base

A *bus base* is a mounting device that is designed to seat an electronic module on a DIN rail and connect it to the TM5 bus for M258 and LMC058 controllers. Each base bus extends the TM5 data and to the power buses and the 24 Vdc I/O power segment. The electronic modules are added to the TM5 system through their insertion on the base bus. The base bus also supplies the articulation point for the terminal blocks.

BYTE

When 8 bits are grouped together, they are called a **BYTE**. You can enter a **BYTE** either in binary mode or in base 8. The **BYTE** type is encoded in an 8-bit format that ranges from 16#00 to 16#FF (in hexadecimal format).

C**calibration**

Graduates a piece of measuring apparatus.

CAN

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

CANmotion

CANmotion is a CANopen-based motion bus with an additional mechanism that provides synchronization between the motion controller and the drives.

CANopen

CANopen is an open industry-standard communication protocol and device profile specification.

CFC

The *continuous function chart* (an extension of the IEC61131-3 standard) is a graphical programming language that works like a flowchart. By adding simple logic blocks (AND, OR, etc.), each function or function block in the program is represented in this graphical format. For each block, the inputs are on the left and the outputs on the right. Block outputs can be linked to inputs of other blocks in order to create complex expressions.

CiA

CAN in automation is a non-profit group of manufacturers and users dedicated to developing and supporting CAN-based higher layer protocols.

CIP

When the *common industrial protocol* is implemented in a network application layer, it can communicate seamlessly with other CIP-based networks without regard to the protocol. For example, the implementation of CIP in the application layer of an Ethernet TCP/IP network creates an EtherNet/IP environment. Similarly, CIP in the application layer of a CAN network creates a DeviceNet environment. In that case, devices on the EtherNet/IP network can communicate with devices on the DeviceNet network through CIP bridges or routers.

CMU

The *current measurement unit* is used to convert the relative current value (%) provided by TeSys into a real ISO value (A).

configuration

The *configuration* includes the arrangement and interconnection of hardware components within a system and the hardware and software selections that determine the operating characteristics of the system.

controller

A *controller* (or “programmable logic controller,” or “programmable controller”) is used to automate industrial processes.

controller status output

The *controller status output* is a special function used in circuits that are external to the controller that control the power supply to the output devices or the controller power supply.

CPDM

controller power distribution module

CRC

A network message’s *cyclic redundancy check* field contains a small number of bits that produce a checksum. The message is calculated by the transmitter according to the message’s content. Receiving nodes then recalculate the field. Any discrepancy in the two CRC fields indicates that the transmitted message and the received message are different.

CSA

The *canadian standards association* defines and maintains standards for industrial electronic equipment in hazardous environments.

CTS

Clear to send is a data transmission signal and acknowledges the RDS signal from the transmitting station.

cyclic task

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

D**data log**

The controller logs events relative to the user application in a data log.

DCE

Data communications equipment describes devices (often modems) that start, stop, and sustain network sessions.

Derating

Derating describes a reduction in an operating specification. For devices in general it is usually a specified reduction in nominal power to facilitate operation at increased ambient conditions like higher temperatures or higher altitudes.

DHCP

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

digital I/O

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

DIN

Deutsches Institut für Normung is a German institution that sets engineering and dimensional standards.

DINT

A *double integer* type is encoded in a 32-bit format.

DNS

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

drop cable

A *drop cable* is the unterminated derivation cord used to connect a TAP to a device.

DSR

Data set ready is a data transmission signal.

DTM

With *device type managers* representing the field device in SoMachine, direct communications are possible to every single field device via SoMachine, the controller and the field bus, thus avoiding the need for individual cable connections.

DWORD

A *double word* type is encoded in a 32-bit format.

E

EDS

Electronic data sheet contains for example the properties of a device e.g. parameters and settings of a drive.

EEPROM

Electrically erasable programmable read-only memory is a type of non-volatile memory used to store data that must be saved when power is removed.

EIA

The *electronic industries alliance* is the trade organization for establishing electrical/electronic and data communication standards (including RS-232 and RS-485) in the United States.

EIA rack

An *electronic industries alliance rack* is a standardized (EIA 310-D, IEC 60297 and DIN 41494 SC48D) system for mounting various electronic modules in a stack or rack that is 19 inches (482.6 mm) wide.

electronic module

In a programmable controller system, most electronic modules directly interface to the sensors, actuators, and external devices of the machine/process. This electronic module is the component that mounts in a bus base and provides electrical connections between the controller and the field devices. Electronic modules are offered in a variety of signal levels and capacities. (Some electronic modules are not I/O interfaces, including power distribution modules and transmitter/receiver modules.)

EN

EN identifies one of many European standards maintained by CEN (*European Committee for Standardization*), CENELEC (*European Committee for Electrotechnical Standardization*), or ETSI (*European Telecommunications Standards Institute*).

encoder

An *encoder* is a device for length or angular measurement (linear or rotary encoders).

Equipment

An *Equipment* is a part of the *Machine*.

ERC

eccentric roller conveyor

ESD

electrostatic discharge

Ethernet

Ethernet is a physical and data link layer technology for LANs, also known as IEE 802.3.

EtherNet/IP

The *ethernet industrial protocol* is an open communications protocol for manufacturing automation solutions in industrial systems. EtherNet/IP is in a family of networks that implements Common Industrial Protocol at its upper layers. The supporting organization (ODVA) specifies EtherNet/IP to accomplish global adaptability and media independence.

expansion bus

The *expansion bus* is an electronic communication bus between expansion modules and a CPU.

expansion I/O module

An *expansion input or output module* is either a digital or analog module that adds additional I/O to the base controller.

expert I/O

Expert I/Os are dedicated modules or channels for advanced features. These features are generally embedded in the module in order to not use the resources of the PLC Controller and to allow a fast response time, depending of the feature. Regarding the function, it could be considered as a “stand alone” module, because the function is independent of the Controller processing cycle, it just exchanges some information with the Controller CPU.

F

FAST I/O

FAST I/Os are specific I/Os with some electrical features (response time, for example) but the treatment of these channels is done by the Controller CPU.

FAST task

The *FAST task* is a periodic, high-priority task of a short duration that is run on a processor through its programming software. The task fast speed keeps it from interfering with the execution of lower priority master (MAST) tasks. A FAST task is useful when fast periodic changes in discrete inputs need to be monitored.

FB

A *function block* performs a specific automation function, such as speed control, interval control, or counting. A function block comprises configuration data and a set of operating parameters.

FBD

A *function block diagram* is a graphically oriented programming language, compliant with IEC 61131-3. It works with a list of networks whereby each network contains a graphical structure of boxes and connection lines which represents either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

FDT

Field device tool for standardized communications between field devices and SoMachine.

FE

Functional ground is the point of a system or device that must be grounded to help prevent equipment damage.

FG

frequency generator

firmware

The *firmware* represents the operating system on a controller.

Flash memory

Flash memory is nonvolatile memory that can be overwritten. It is stored on a special EEPROM that can be erased and reprogrammed.

FTP

File transfer protocol is a standard network protocol (built on a client-server architecture), to exchange and manipulate files over TCP/IP based networks.

function

A *function*:

- is a POU that returns 1 immediate result
- is directly called with its name (as opposed to through an instance)
- has no persistent state from one call to the next
- can be used as an operand in expressions

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

function block (FB)

See *FB*.

function block diagram (FBD)

See *FBD*.

FWD

forward

G

gross weight

Indication of the load weight on an instrument when no tare or predefining device has been used.

GVL

The *global variable list* manages global variables that are available in every application POU.

H

HE10

Rectangular connector for electrical signals with frequencies below 3MHz, complying with IEC60807-2.

HMI

A *human-machine interface* is an operator interface (usually graphical) for industrial equipment.

hot swapping

Hot swapping is the replacement of a component with a like component while the system remains operational. The replacement component begins to function automatically after it is installed.

HSC

high-speed counter

HVAC

Heating ventilation and air conditioning applications monitor and control indoor environments.

I**I/O**

input/output

I/O scan

An *input/output scan* continuously polls I/O modules to collect data bits and status, error, and diagnostics information. This process monitors inputs and controls outputs.

I/O terminal

An *input/output terminal* on the front of an expansion I/O module connects input and output signals.

ICMP

The *internet control message protocol* reports errors and provides information related to datagram processing.

IEC

The *international electrotechnical commission* is a non-profit and non-governmental international standards organization that prepares and publishes international standards for all electrical, electronic, and related technologies.

IEC 61131-3

The IEC 61131-3 is an *international electrotechnical commission* standard for industrial automation equipment (like controllers). IEC 61131-3 deals with controller programming languages and defines 2 graphical and 2 textual programming language standards:

- **graphical:** ladder diagram, function block diagram
- **textual:** structured text, instruction list

IEEE

The *institute of electrical and electronics engineers* is a non-profit international standards and conformity assessment body for advances in all fields of electrotechnology.

IEEE 802.3

IEEE 802.3 is a collection of IEEE standards defining the physical layer, and the media access control (MAC) sublayer of the data link layer, of wired Ethernet.

IL

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

immediate addressing

The direct method of addressing memory objects, including physical inputs and outputs, used in programming instructions as operands and parameters by using their direct address (for example, %Iwx or %QWx).

The use of immediate addressing in your program may avoid the need to create symbols for these objects, but there are also disadvantages. For example, if you change the program configuration by adding or deleting devices or I/O modules or slices, the immediate addresses used as programming instruction operands and/or parameters are not updated and must be corrected manually, which may cause extensive program modifications and lead to incorrect programming instructions. (See *symbolic addressing*.)

input filter

An *input filter* is a special function that rejects input noises. It is useful for eliminating input noises and chatter in limit switches. All inputs provide a level of input filtering using the hardware. Additional filtering with software is also configurable through the programming or the configuration software.

input terminal

An *input terminal* on the front of an expansion I/O module connects input signals from input devices (such as sensors, push buttons, and limit switches). For some modules, input terminals accept both sink and source DC input signals.

instruction list language (IL)

Refer to IL.

INT

A single *integer* is encoded in 16 bits.

IP

The *internet protocol* is part of the TCP/IP protocol family that tracks the Internet addresses of devices, routes outgoing messages, and recognizes incoming messages.

IP 20

Ingress protection rating according to IEC 60529. IP20 modules are protected against ingress and contact of objects larger than 12.5 mm. The module is not protected against harmful ingress of water.

IP 67

Ingress protection rating according to IEC 60529. IP67 modules are completely protected against ingress of dust and contact. Ingress of water in harmful quantity is not possible when the enclosure is immersed in water up to 1m.

K**Kd**

derivative gain

Ki

integral gain

Kp

proportional gain

L**Ladder Diagram language**

See *LD*.

LAN

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

latching input

A *latching input* module interfaces with devices that transmit messages in short pulses. Incoming pulses are captured and recorded for later examination by the application.

LCD

liquid crystal display

LD

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

LED

A *light emitting diode* is an indicator that lights up when electricity passes through it.

LINT

Long integer is a 64-bit variable (4 times INT or two times DINT).

LMC

lexium motion control

load receiver device

Part of instrument that will receive the load.

located variable

A *located variable* has an address. (See *unlocated variable*.)

LRC

longitudinal redundancy checking

LREAL

Long real is a 64-bit variable.

LSB

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

LWORD

A *long word* type is encoded in a 64-bit format.

M**MAC address**

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

Machine

A *Machine* consists of several *functions* and/or *equipments* which build the machine.

Magelis

Magelis is the commercial name for Schneider Electric's range of HMI terminals.

MAST

A master (MAST) task is a processor task that is run through its programming software. The MAST task has two sections:

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

master/slave

The single direction of control in a network that implements the master/slave model is always from a master device or process to one or more slave devices.

maximum weight

Maximum measuring capacity, not taking account of the additive capacity of the tare.

MIB

The *management information base* is an object database that is monitored by a network management system like SNMP. SNMP monitors devices that are defined by their MIBs. Schneider has obtained a private MIB, *groupeschneider* (3833).

minimum I/O update time

The *minimum I/O update time* is the minimum time it takes for the bus cycle to shut down to force an I/O update at each cycle.

minimum weight

Load value under which measuring results can be marred by a relative detected error that is too large.

Modbus

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

Modbus SL

Modbus serial line

MSB

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

N

NAK

negative acknowledge

NC

A *normally closed* contact is a contact pair that is closed when the actuator is de-energized (no power is applied) and open when the actuator is energized (power is applied).

NEC

The *national electric code* dictates the safe installation of electrical wiring and equipment.

NEMA

The *national electrical manufacturers association* publishes standards for the performance of various classes of electrical enclosures. The NEMA standards cover corrosion resistance, ability to protect from rain and submersion, etc. For IEC member countries, the IEC 60529 standard classifies the ingress protection rating for enclosures.

net weight (net)

Weight indication of a load placed on an instrument after a tare device has been used.

Net weight = Gross weight - Tare weight

network

A network includes interconnected devices that share a common data path and protocol for communications.

Nibble

A *Nibble* is a half-byte (representing 4 bits of a byte).

NMT

Network management protocols provide services for network initialization, error control, and device status control.

NMT state machine

A *network management state machine* defines the communication behavior of any CANopen device. The CANopen NMT state machine consists of an initialization state, a pre-operational state, an Operational state, and a stopped state. After power-on or reset, the device enters the initialization state. After the device initialization is finished, the device automatically enters the pre-operational state and announces the state transition by sending the boot-up message. In this manner, the device indicates that it is ready to work. A device that stays in pre-operational state may start to transmit SYNC-, Time Stamp-, or Heartbeat message. In this state, the device can not communicate through a PDO; it must do so with an SDO. In the operational state, the device can use all supported communication objects.

NO

A *normally open* contact is a contact pair that is open when the actuator is de-energized (no power is applied) and closed when the actuator is energized (power is applied).

node

A *node* is an addressable device on a communication network.

O

ODVA

The *open deviceNet vendors association* supports the family of network technologies that are built on CIP (EtherNet/IP, DeviceNet, and CompoNet).

OS

Operating system. Can be used for Firmware that can be uploaded/downloaded by the user.

OSI

The *open system interconnection* reference model is a 7-layer model that describes network protocol communications. Each abstract layer receives services from the layer below it and provides services to the layer above.

OTB

Optimized terminal block, used in the context of Advantys I/O distributed module

output terminal

An *output terminal* connects output signals to output devices (such as electromechanical relays and solenoid valves).

P

pallet

A *pallet* is a portable platform, which is used for storing or moving goods.

PCI

A *peripheral component interconnect* is an industry-standard bus for attaching peripherals.

PDM

A *power distribution module* distributes either AC or DC field power to a cluster of I/O modules.

PDO

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

PDU

protocol data unit

PE

Protective ground is a return line across the bus for fault currents generated at a sensor or actuator device in the control system.

periodic execution

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

persistent data

Value of persistent data that will be used at next application change or cold start. Only get re-initialized at a reboot of the controller or reset origin. Especially they maintain their values after a download.

PI

proportional integral

PID

proportional, integral and derivative control

PLC

The *programmable logic controller* is the “brain” of an industrial manufacturing process. It automates a process, used instead of relay control systems. PLCs are computers suited to survive the harsh conditions of the industrial environment.

PLCopen

The PLCopen standard brings efficiency, flexibility, and manufacturer independence to the automation and control industry through the standardization of tools, libraries, and modular approaches to software programming.

PLI

pulse latch input

post-configuration

Post-configuration files contain machine-independent parameters, including:

- machine name
- device name or IP address
- Modbus serial line address
- routing table

POU

A *program organization unit* includes a variable declaration in source code and the corresponding instruction set. POU's facilitate the modular reuse of software programs, functions, and function blocks. Once declared, POU's are available to one another. SoMachine programming requires the utilization of POU's.

POU FB

Program organization unit function block types are user programs that can be defined by the user in the ST, IL, LD, or FBD languages. You can use POU FB types in an application to:

- simplify the design and entry of the program
- make the program easier to read
- simplify debugging
- reduce the amount of generated code

power supply terminals

The power supply is connected to these terminals to provide power to the controller.

Profibus DP

Profibus Decentralized Peripheral

An open bus system that uses an electrical network based on a shielded 2-wire line or an optical network based on a fiber-optic cable. DP transmission allows for high-speed, cyclic exchange of data between the controller CPU and the distributed I/O devices.

protocol

A *protocol* is a convention or standard that controls or enables the connection, communication, and data transfer between two computing endpoints.

Pt100/Pt1000

Platinum resistance thermometer are characterized by their nominal resistance R_0 at a temperature of 0°C .

- Pt100 ($R_0 = 100 \text{ Ohm}$)
- Pt1000 ($R_0 = 1 \text{ kOhm}$)

PTO

Pulse train outputs are used to control for instance stepper motors in open loop.

PWM

Pulse width modulation is used for regulation processes (e.g. actuators for temperature control) where a pulse signal is modulated in its length. For these kind of signals, transistor outputs are used.

R**RAM**

random access memory

REAL

Real is a numeric data type. The REAL type is encoded in a 32-bit format.

real-time clock (RTC)

See RTC

reflex output

In a counting mode, the high speed counter current value is measured against its configured thresholds to determine the state of these dedicated outputs.

retained data

A *retained data* value is used in the next power-on or warm start. The value is retained even after an uncontrolled shutdown of the controller or a normal switch-off of the controller.

RFID

Radio-frequency identification is an automatic identification method that relies on the storage and remote retrieval of data using RFID tags or transponders.

RJ-45

This *registered jack* is a modular connector that is commonly implemented in communication networks.

RPDO

A *receive PDO* sends data to a device in a CAN-based network.

RPM

revolutions per minute

RPS

revolutions per second

RS-232

RS-232 (also known as EIA RS-232C or V.24) is a standard type of serial communication bus, based on three wires.

RS-485

RS-485 (also known as EIA RS-485) is a standard type of serial communication bus, based on two wires.

RTC

The *real-time clock* option keeps the time for a limited amount of time even when the controller is not powered.

RTS

Request to send is a data transmission signal and will be acknowledged by the CTS signal from the destination node.

RTU

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

RxD

receiving data (data transmission signal)

S**SCADA**

A *supervisory control and data acquisition* system monitors, manages, and controls industrial applications or processes.

scale division

Value in mass units, expressing the difference between two consecutive indications for one numerical indication.

scan

A controller scanning program performs 3 basic functions: [1] It reads inputs and places these values in memory; [2] it executes the application program 1 instruction at a time and stores results in memory; [3] It uses the results to update outputs.

SDO

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

SEL-V

A system that follows IEC 61140 guidelines for *safety extra low voltage* is protected in such a way that voltage between any 2 accessible parts (or between 1 accessible part and the PE terminal for Class 1 equipment) does not exceed a specified value under normal conditions or under single-fault conditions.

Sequential Function Chart

See *SFC*.

SFC

A program written in the *sequential function chart* language can be used for processes that can be split into steps. SFC is composed of steps with associated actions, transitions with associated logic condition, and directed links between steps and transitions. (The SFC standard is defined in IEC 848. It is IEC 61131-3 compliant.)

sink input

A *sink input* is a wiring arrangement in which the device provides current to the input electronic module. A sink input is referenced to 0 Vdc.

SINT

Signed integer is a 16-bit value.

SL

serial line

SMS

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

SNMP

The *simple network management protocol* can control a network remotely by polling the devices for their status, performing security tests, and viewing information relating to data transmission. It can also be used to manage software and databases remotely. The protocol also permits active management tasks, such as modifying and applying a new configuration

source output

A *source output* is a wiring arrangement in which the output electronic module provides current to the device. A source output is referenced to +24 Vdc.

SSI

Serial synchronous interface is a common interface for relative and absolute measurement systems like encoders.

ST

See *structured text*.

STN

Scan Twisted Nematic (also known as passive matrix)

STRING

A **STRING** variable is a series of ASCII characters.

Structured Text

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

symbol

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

symbolic addressing

The indirect method of addressing memory objects, including physical inputs and outputs, used in programming instructions as operands and parameters by first defining symbols for them using these symbols in association with the programming instructions.

In contrast to immediate addressing, this is the recommended method because if the program configuration changes, symbols are automatically updated with their new immediate address associations, whereas any immediate addresses used as operands or parameters are not. (See *immediate addressing*.)

system time

An internal clock provides a device with the system time.

system variable

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

T**TAP**

A *terminal access point* is a junction box connected to the trunk cable that allows you to plug in drop cables.

tare

Load placed on the load receiver along with the product to be weighed.

tare device

Device allowing the instrument indication to be moved to zero when a load is positioned on the load receiver:

tare predefining device

Device allowing a predefined tare value to be subtracted from a gross weight value and indicating the result of the calculation. The load range is consequently reduced.

Tare Value

Weight value of a load, determined by a tare full-bridge strain gauge electronic module.

taring

Action allowing the instrument indication to be moved to zero when a load is positioned on the load receiver.

task

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

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

A controller can have several tasks.

TCP

A *transmission control protocol* is a connection-based transport layer protocol that provides a reliable simultaneous bi-directional transmission of data. TCP is part of the TCP/IP protocol suite.

terminal block

The *terminal block* is the component that mounts in an electronic module and provides electrical connections between the controller and the field devices.

TFT

thin film transmission (also known as active matrix)

threshold output

Threshold outputs are controlled directly by the HSC according to the settings established during configuration.

TP

A *touch probe* is a position capture that is triggered by a fast input signal (quick sensor). On the rising edge of the touch probe input the position of an encoder is captured. Example: This is used for packaging machines to capture the position of a printmark on a film to cut always on the same position.

TPDO

A *transmit PDO* reads data from a device in a CAN-based system.

trunk cable

A *trunk cable* is the main cable that is terminated at both physical ends with line termination resistors.

TVDA

tested validated documented architectures

TxD

TxD represents a transmit signal.

U**UDINT**

An *unsigned double integer* is encoded in 32 bits.

UDP

The *user datagram protocol* is a connectionless mode protocol (defined by IETF RFC 768) in which messages are delivered in a datagram (data telegram) to a destination computer on an IP network. The UDP protocol is typically bundled with the Internet Protocol. UDP/IP messages do not expect a response, and are therefore ideal for applications in which dropped packets do not require retransmission (such as streaming video and networks that demand real-time performance).

UINT

An *unsigned integer* is encoded in 16 bits.

UL

Underwriters Laboratories, US organization for product testing and safety certification.

unlocated variable

An *unlocated variable* does not have an address. (See *located variable*.)

UTC

coordinated universal time

V

VSD

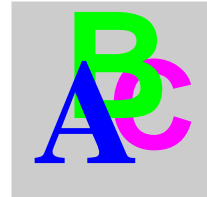
variable speed drive

W

WORD

The *WORD* type is encoded in a 16-bit format.

Index



A

- adjusting functions
 - EXPERTGetParam, 196
 - EXPERTSetParam, 198

B

- Busy
 - Management of Status Variables, 181

C

- Capture
 - Encoder, 166
 - HSCMain, 163
- CommandAborted
 - Management of Status Variables, 181
- Comparison
 - HSCMain, 154
- Counter Blocks
 - HSCMain, 203
 - HSCSimple, 208

D

- Data Types
 - EXPERT_ERR_TYPE, 185
 - EXPERT_PARAMETER_TYPE, 186
 - EXPERT_TIMEBASE_TYPE, 188
 - HSC_REF, 187
 - MC_IMMEDIATE_ERR_TYPE, 184
- Dedicated Functions, 180

- diagnostic functions

 - EXPERTGetDiag, 193

- Done

 - Management of Status Variables, 181

E

- Enable
 - Function, 176
- Encoder
 - Capture, 166
 - Function Blocks, 200
- Encoder Modes
 - Incremental, 136
- ErrID
 - Error Handling, 181
 - Management of Status Variables, 181
- Error
 - Error Handling, 181
 - Management of Status Variables, 181
- Error Handling
 - ErrID, 181
 - Error, 181
- Event Counting
 - HSC Modes of Embedded HSC, 97
- Execute
 - Management of Status Variables, 181
- EXPERT_ERR_TYPE
 - Data Types, 185
- EXPERT_PARAMETER_TYPE
 - Data Types, 186

EXPERT_TIMEBASE_TYPE
Data Types, 188
EXPERTGetCapturedValue
Function Blocks, 191
EXPERTGetDiag
Function Blocks, 193
EXPERTGetImmediateValue
Functions, 190
EXPERTGetParam
Function Blocks, 196
EXPERTSetParam
Function Blocks, 198

F

Free-large
HSC Modes of Embedded HSC, 80
Frequency Meter
HSC Modes of Embedded HSC, 111
Function
Enable, 176
Function Blocks
Encoder, 200
EXPERTGetCapturedValue, 191
EXPERTGetDiag, 193
EXPERTGetParam, 196
EXPERTSetParam, 198
HSCMain, 203
HSCSimple, 208
Functions
Differences Between a Function and a
Function Block, 212
EXPERTGetImmediateValue, 190
How to Use a Function or a Function
Block in IL Language, 213
How to Use a Function or a Function
Block in ST Language, 216

H

HSC main type configuration
Event mode, 101
Free-large mode, 87
HSC Main type configuration
Frequency Meter mode, 115

HSC main type configuration
modulo loop mode, 141
Modulo-loop mode, 69
One-shot mode, 47
Period Meter mode, 127
HSC Modes of Embedded HSC
Event Counting, 97
Free-large, 80
Frequency Meter, 111
Modulo-loop, 57
HSC Modes of HSC
Period Meter, 123
HSC simple type configuration
Modulo-loop mode, 61
One-shot mode, 39
HSC_REF
Data Types, 187
HSCMain
Capture, 163
Comparison, 154
Function Blocks, 203
HSCSimple
Function Blocks, 208

I

Incremental
Encoder Modes, 136

M

Management of Status Variables
Busy, 181
CommandAborted, 181
Done, 181
ErrID, 181
Error, 181
Execute, 181
MC_IMMEDIATE_ERR_TYPE
Data Types, 184
Modulo-loop
HSC Modes of Embedded HSC, 57

P

Period Meter

HSC Modes of HSC, 123

