Modicon LMC058 Motion Controller

High Speed Counting
LMC058 Expert I/O Library Guide

09/2020

EIO0000004177.00

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.

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

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

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

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

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

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

This documentation will acquaint you with the High Speed Counter (HSC) and Encoder functions and variables offered within the LMC058 controller.

This documentation describes the functions and variables of the LMC058 HSC and Encoder library.

In order to use this manual, you must:

- Have a thorough understanding of the LMC058, 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)

Only DM72F• expert module and Encoder module can be used with the LMC058 HSC and Encoder library.

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.2.5.

The technical characteristics of the devices described in the present document also appear online. To access the information online, go to the Schneider Electric home page *https://www.se.com/ww/en/download/*.

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

Related Documents

Title of Documentation	Reference Number
Modicon LMC058 Logic Controller Programming Guide	*EIO0000004165 (Eng)* *EIO0000004166 (Fre)* *EIO0000004167 (Ger)* *EIO0000004168 (Spa)* *EIO0000004169 (Ita)* *EIO0000004170 (Chs)*
Modicon LMC058 Logic Controller Hardware Guide	*EIO0000004189 (Eng)* *EIO0000004190 (Fre)* *EIO0000004191 (Ger)* *EIO0000004192 (Spa)* *EIO0000004193 (Ita)* *EIO0000004194 (Chs)*

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

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Terminology Derived from Standards

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

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

Among others, these standards include:

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

Standard	Description
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

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

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

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

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

Part I
High Speed Counter and Encoder Overview

Overview

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

What Is in This Part?

This part contains the following chapters:

Chapter 1
Introduction

What Is in This Chapter?

This chapter contains the following topics:

Expert I/O Overview

Introduction

The controller base provides:

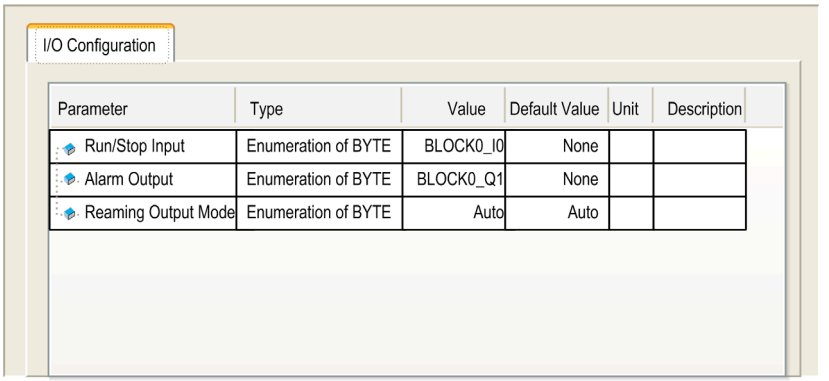
- 2 embedded expert I/O modules (DM72F0 and DM72F1) with:
  - 5 fast inputs
  - 2 regular inputs
  - 2 fast outputs
- 1 Hardware Encoder port that can support:
  - Incremental encoder
  - SSI absolute encoder
- 1 Controller Power Distribution Module (CPDM)

Each embedded expert I/O module (DM72F•) can support expert functions *(see page 21)*.

Embedded Expert I/O Configuration

To configure the expert I/Os, double-click the **Expert** node in the **Devices tree**.

This figure presents the configuration tab screenshot:



This table presents the function of the different parameters:

Parameter	Function
Run/Stop Input	Define one input to be used as Run/Stop input *(see page 19)*.
Alarm Output	Define one output to be used as alarm output *(see page 19)*.
Rearming Output Mode	Define the rearming output mode *(see page 20)*.

Run/Stop Input

This table presents the different states:

Input states	Result
State 0	Stops the controller and ignores external Run commands.
A rising edge	From the STOPPED state, initiates a start-up of an application in RUNNING state.
State 1	The application can be controlled by: ● EcoStruxure Machine Expert (Run/Stop) ● the application (Controller command) ● a network command

NOTE: Run/Stop input is managed even if the option **Update IO while in stop** is not selected in the PLC settings tab.
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.

⚠ WARNING

UNINTENDED MACHINE OR PROCESS START-UP

- Verify the state of security of your machine or process environment before applying power to the Run/Stop input.
- Use the Run/Stop input to help prevent the unintentional start-up from a remote location.

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

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 I/O (power interruption, short-circuit detection).

Rearming Output Mode

Fast outputs of DM72F• modules are push/pull technology. In the case of an error detected (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 (refer to the Modicon LMC058 Motion Controller System Functions and Variables PLCSystem 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 enters into thermal protection mode (all outputs in the group are set to 0), and are then periodically rearmed (each second) to test the connection state. However, you must be aware of the effect of this rearming on the machine or process being controlled.

⚠ 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:
  - High Speed Counter Simple
  - Event_Latch I/O
- complex functions:
  - High Speed Counter Main
  - Encoder
  - Frequency Generator (FreqGen)
  - Pulse Width Modulation (PWM)

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 18)*.

Adding an Expert Function

To add an Expert function (Event_Latch, HSC, PWM or Frequency Generator) to your controller, select the Expert function you want in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

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

• Using the Drag-and-drop Method

• Using the Contextual Menu or Plus Button

To add an Encoder function, select the **Standard Encoder** in the **Hardware Catalog**, drag it to the **Devices tree**, and drop it on one of the highlighted nodes.

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
HSC	The HSC functions can execute fast counts of pulses from sensors, encoders, switches, etc. that are connected to dedicated fast inputs.	LMC058 HSC Library
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%).	LMC058 PWM library
Encoder	The goal of this function is to connect an encoder to acquire a position. This function can be implemented on an Embedded Expert I/O interface and an Hardware Encoder interface. The Encoder can be Incremental or absolute SSI on an Hardware Encoder interface. The Embedded Expert I/O interface supports only an Incremental Encoder. You can configure a linear or rotary axis for incremental encoder.	LMC058 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	SM_Encoder	Encoder	PWM	Frequency Generator
DM72F0	Up to 4	1	1	1	1	1
DM72F1	Up to 4	1	1	1	1	1
Encoder	Not allowed	Not allowed	1	1	Not allowed	Not allowed

For more details, refer to Expert I/O Mapping *(see page 24)*.

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 an 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 (refer to the Modicon LMC058 Motion Controller Hardware Guide). Filter value will be configured in expert function screen.

Embedded Expert I/O Mapping

I/O Mapping for Expert Functions on DM72F•

Embedded Expert I/O mapping by expert function:

		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

M Mandatory
C Depends on Configuration

		I0	I1	I2	I3	I4	I5	Q0	Q1
Motion Encoder	Input A	M							
	Input B		M						
	Input Z			M					
	CAP				C				
M Mandatory **C** Depends on Configuration									

NOTE: The DM72F• I6 inputs can only be configured by encoder on ENC.

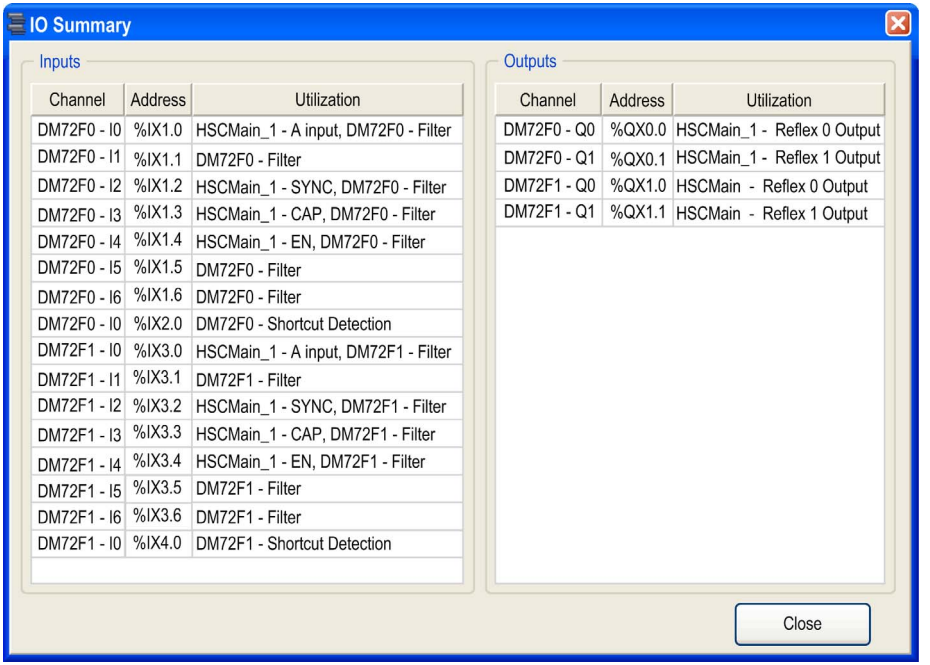
IO Summary

The **IO Summary** window displays the DM72F• I/O and the I/O used by expert functions.

The **IO Summary** window is accessible from **DM72F•** nodes:

Step	Action
1	In the **Devices tree** tab, expand the **Expert** node.
2	Right-click **DM72F•** and select **IO Summary** in context menu.

Example of IO Summary:



Hardware Encoder Interface

Introduction

The controller has a specific hardware encoder interface that can support:

- Incremental encoder
- SSI absolute encoder

Encoder function

The goal of this function is to connect an encoder to acquire a position so that it can be used as Master Axis for motion drives on CAN.

This function can be implemented on an Embedded Expert I/O interface and a hardware Encoder interface. The Encoder can be Incremental or absolute SSI on a hardware Encoder interface. The Embedded Expert I/O interface supports only an Incremental Encoder.

You can configure a linear or rotary axis for incremental encoder.

I/O mapping

Input of Embedded Expert I/O modules (DM72F•) used by standard and motion encoder function:

	DM72F0 I6	DM72F1 I6	Encoder type
CAP0	X	–	Standard Motion
CAP1	–	X	Standard Motion
EN	X	–	Standard
REF	–	X	Standard
X Depends on configuration			

Chapter 2
High Speed Counter Overview

Overview

This chapter provides a main description of the different types of High Speed Counters.

What Is in This Chapter?

This chapter contains the following topics:

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 on single-phase pulses:

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

Modulo-loop *(see page 61)*. 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 200 kHz.

For more information about the bounce filter, refer to Dedicated Features *(see page 202)*.

Main Type Overview

Overview

The **Main** type is a counter that uses up to 6 fast inputs and 2 reflex outputs.

Main Type Modes

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

One-shot *(see page 47)*: In this mode, the counter value register decrements (from a user-defined value) for each pulse applied to the A input until the counter reaches 0.

Modulo-loop *(see page 67)*: In this mode, the counter repeatedly counts up from 0 to a user-defined modulo value, then returns to 0 and restarts counting. In reverse, the counter counts down from the modulo value to 0, then presets to the modulo value and restarts counting.

Free-large *(see page 83)*: In this mode, the counter behaves like a full range up and down counter. Can be used with one encoder.

Event Counting *(see page 95)*: In this mode, the counter accumulates the number of events that are received during a user-configured time base.

Frequency meter *(see page 107)*: In this mode, the counter measures the frequency of events. Frequency is the number of events per second (Hz).

Period meter *(see page 119)*: Use the **Period meter** mode to:
- determine the duration of an event
- determine the time between 2 events
- set and measure the execution time for a process

Optional Features

Optional features can be configured depending on the selected mode:
- Hardware inputs to operate the counter (enable, preset) or capture the on-going counting value
- Up to 4 thresholds, the values of which can be compared.
- Up to 4 events (1 for each 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

Chapter 3
Encoder Overview

Overview

This chapter provides a general overview of the encoders.

What Is in This Chapter?

This chapter contains the following topics:

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) and the **Encoder** interface (Sub-D).

Standard Encoder Modes

When implemented on an **Expert I/O** interface, the encoder only supports the Incremental *(see page 133)* mode.

When implemented on the **Encoder** interface, the encoder has 2 possible modes:

- Incremental *(see page 142)*
- absolute SSI *(see page 162)*

On the **Encoder** interface, the power supply is monitored. A power supply error detected 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

Maximum admissible frequency:

- **Expert I/O** module: 100 kHz.
- **Encoder** interface: 200 kHz.

Motion Encoder Overview

General

The goal of this function is to connect an encoder to acquire a position. This function can be used as Master Axis for motion drives on CANmotion.

This function can be implemented on the **Expert I/O** module (DM72F0 and DM72F1) and the **Encoder** interface (Sub-D).

The Motion Encoder configuration is split in 2 parts:
- A hardware part: Motion Encoder; described in this document.
- A software part: SoftMotion encoder; described in the online help CoDeSys part, Editors/Devices Editors/SoftMotion Device Editor.

The Motion Encoder is managed by function blocks from the LMC058 Motion library and SoftMotion (See online help Programming with EcoStruxure Machine Expert part, SoftMotion/Programming Interface/SoftMotion Libraries/SM3_Basic_library).

Motion Encoder Modes

When implemented on an **Expert I/O** interface, the encoder support the Incremental *(see page 151)* mode.

When implemented on the Encoder interface, the encoder has 2 possible modes:
- Incremental *(see page 155)*
- absolute SSI *(see page 169)*

On the Encoder interface, the power supply is monitored. A power supply error detected (SMC_DI_VOLTAGE_DISABLED) is cleared automatically when the power supply recovers.

Optional Feature

In addition, the Motion Encoder provides a capture function.

Performance

Maximum admissible frequency:
- **Expert I/O** module: 100 kHz.
- **Encoder** interface: 200 kHz.

Part II
One-shot Mode

Overview

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

What Is in This Part?

This part contains the following chapters:

Chapter 4
One-shot Mode Principle

One-shot Mode Principle Description

Overview

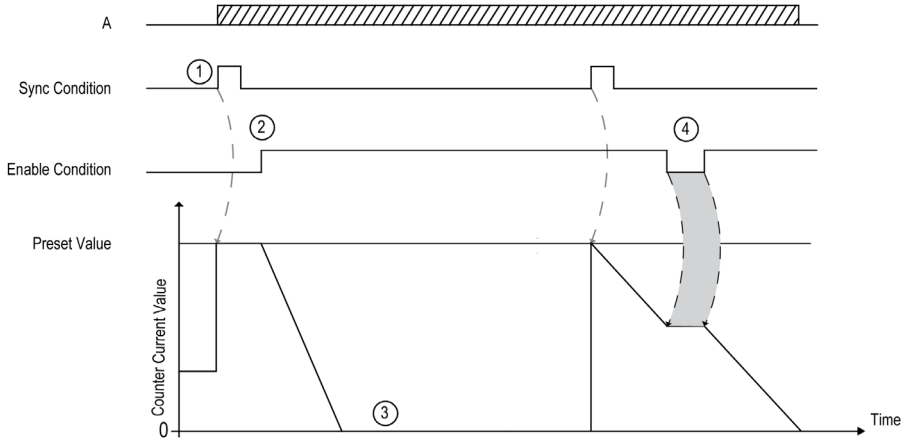
The counter is activated by a synchronization edge, and the preset value is loaded.

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

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

A new synchronization is needed to activate the counter again.

Principle Diagram



This table explains the stages from the preceding graphic:

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

NOTE: Enable and Sync conditions depends on configuration. These are described in the Enable *(see page 198)* and Preset *(see page 194)* function.

Chapter 5
One-shot with a Simple Type

Overview

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

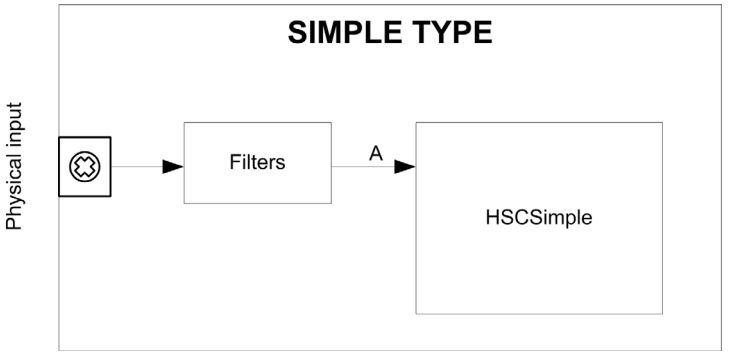
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A is the counting input of the `High Speed Counter`. **Simple** type counting for **One-shot** mode only counts up.

Configuration of the Simple Type in One-shot Mode

Configuration Procedure

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

Step	Action
1	Double-click **Expert → DM72F• → HSCSimple•**.
2	Select the **HSC Simple Configuration** tab.
3	Set the value of the **General → Counting Mode** parameter to **One-Shot**.
4	Select the value of the **Counting inputs → A input → Bounce filter** parameter.
5	Enter the value of the **Range → Preset** parameter.

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

Programmable Filter

The filtering value on the **Simple** 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 (default value)	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

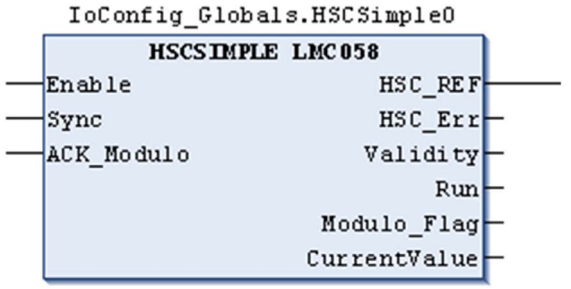
Overview

A **Simple** type counter is always managed by an HSCSimple function block.

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

Adding an HSCSimple Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → HSC → HSCSimple_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Simple** type instance name (defined in configuration) or select the function block instance by clicking: `???` `...` 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.

This table describes the input variables:

Input	Type	Comment
Sync	BOOL	On rising edge, loads the preset of the counter.
ACK_Modulo	BOOL	Not used in one-shot mode.

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	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 226)* 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.
CurrentValue	DWORD	Current value of the counter.
Modulo_Flag	BOOL	Not used in one-shot mode.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 228)* or EXPERTSetParam *(see page 230)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 211)* that can be read or modified while the program is running:

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

Chapter 6
One-shot with a Main Type

Overview

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

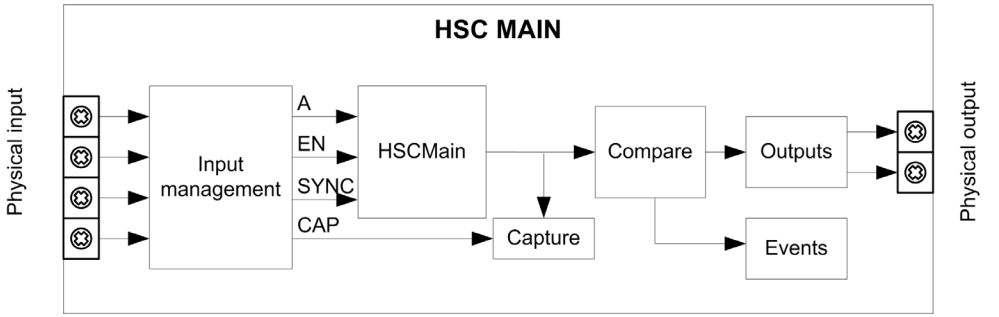
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A is the counting input of the counter.

EN is the enable input of the counter.

SYNC is the synchronization input of the counter.

CAP is the capture input of the counter.

Optional Function

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

- Preset function *(see page 194)*
- Enable function *(see page 198)*
- Capture function *(see page 185)*
- Comparison function *(see page 175)*

Configuration of the Main Type in One-shot Mode

Configuration Procedure

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

Step	Action
1	Double-click **Expert → DM72F• → HSCMain**.
2	Select the **HSC Main Configuration** tab.
3	Set the value of the **General → Counting Mode** parameter to **One-Shot**.
4	Select the value of the **Counting inputs → A input → Bounce filter** parameter.
5	Enter the value of the **Range → Preset** parameter.
6	Optionally, select the value of the **SYNC**, **EN** and **CAP** auxiliary inputs from the from the drop-down menu, to enable the preset function *(see page 194)*, Enable function *(see page 198)* and Capture function *(see page 186)* with a physical input.
7	Optionally, select the **Number of thresholds**. This enables the Compare function and reflex outputs can be configured *(see page 175)*.
8	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 183)*.

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

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 (default value)	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

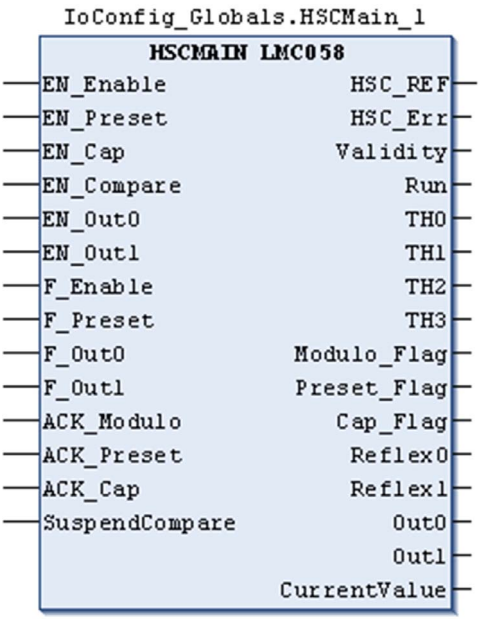
Overview

The **Main** type is always managed by an `HSCMain` function block.

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

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → HSC → HSCMain_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking:  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.

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes enabling of the counter with the Enable input *(see page 198)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter synchronization and start via the Sync input *(see page 194)*.
EN_Cap	BOOL	When **CAP** input is configured: if TRUE, enables the Capture input *(see page 186)*.
EN_Compare	BOOL	TRUE = enables the comparator operation *(see page 175)* (using Thresholds 0, 1, 2, 3): • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1 output bits) • events (to trigger external tasks on threshold crossing) **NOTE:** This option is only available for TM3XF• expansion modules, which support external events.
EN_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 *(see page 198)*. Takes priority over EN_Enable input.
F_Preset	BOOL	Forces the Preset condition *(see page 194)*. Takes priority over EN_Preset input.
F_Out0	BOOL	TRUE = forces Output0 to 1 (if Reflex0 is configured in HSC Embedded Function. Takes priority over EN_Out0.
F_Out1	BOOL	TRUE = forces Output1 to 1 (if Reflex1 is configured in HSC Embedded Function. Takes priority over EN_Out1.
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: • 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.

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	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 226)* 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 175)*.
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 *(see page 175)*.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 *(see page 175)*.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 *(see page 175)*.
Modulo_Flag	BOOL	Set to TRUE when counter reaches 0.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 194)*.
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 186)*. This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0 *(see page 179)*.
Reflex1	BOOL	State of Reflex1 *(see page 179)*.
Out0	BOOL	State of physical output Output0 to 1 (if Reflex0 is configured in HSC Embedded Function, otherwise FALSE if not configured).
Out1	BOOL	State of physical output Output1 to 1 (if Reflex1 is configured in HSC Embedded Function, otherwise FALSE if not configured).
CurrentValue	DINT	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 228)* or EXPERTSetParam *(see page 230)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 211)* which can be read or modified while the program is running:

Parameter	Description
EXPERT_PRESET	to get or set the Preset value of an HSC
EXPERT_THRESHOLD0	to get or set the Threshold 0 value of an HSC
EXPERT_THRESHOLD1	to get or set the Threshold 1 value of an HSC
EXPERT_THRESHOLD2	to get or set the Threshold 2 value of an HSC
EXPERT_THRESHOLD3	to get or set the Threshold 3 value of an HSC
EXPERT_REFLEX0	to get or set output 0 reflex mode of an EXPERT function
EXPERT_REFLEX1	to get or set output 1 reflex mode of an EXPERT function

Part III
Modulo-loop Mode

Overview

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

What Is in This Part?

This part contains the following chapters:

Chapter 7
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 194)*, the counter is activated and the counter value is set to 0.

When counting is enabled *(see page 198)*:

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

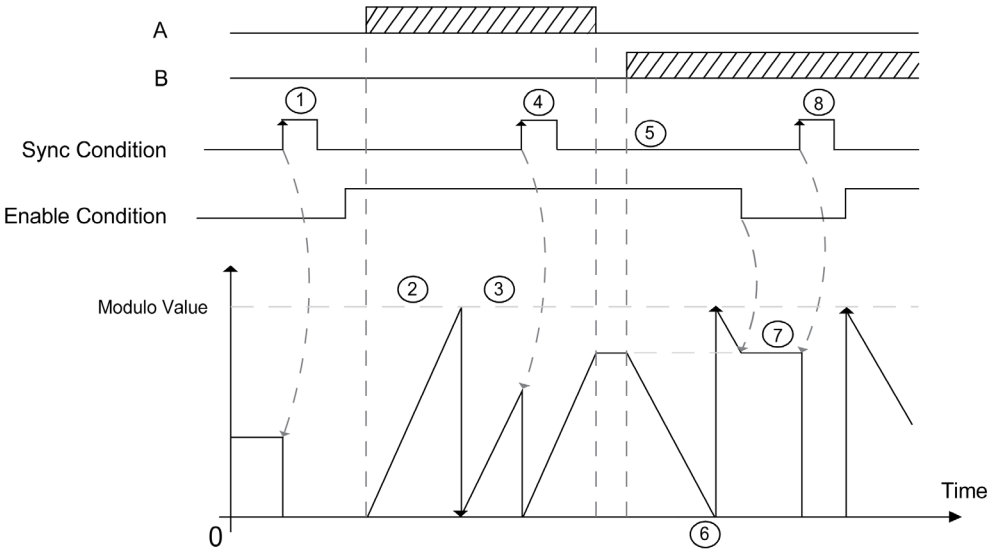
Decrementing direction: the counter decrements until it reaches 0. At the next pulse, the counter is set to the modulo value, a modulo flag is set to 1, and the counting continues.

Input Modes

This table shows the 8 types of input modes available:

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

Up Down Principle Diagram

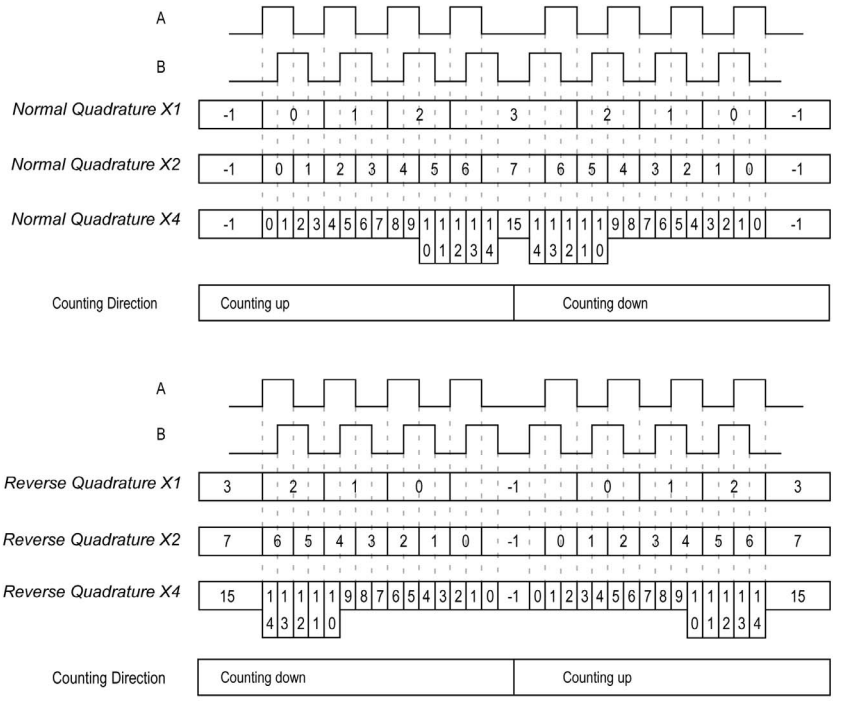


Stage	Action
1	On the rising edge of Sync condition, the counter value is reset to 0 and the counter is activated.
2	When Enable condition = 1, each pulses on A increments the counter value.
3	When the counter reaches the (modulo-1) value, the counter loops to 0 at the next pulse and the counting continues.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

NOTE: Enable and Sync conditions depends on configuration. These are described in the Enable *(see page 198)* and Preset *(see page 194)* function.

Quadrature Principle Diagram

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



Chapter 8
Modulo-loop with a Simple Type

Overview

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

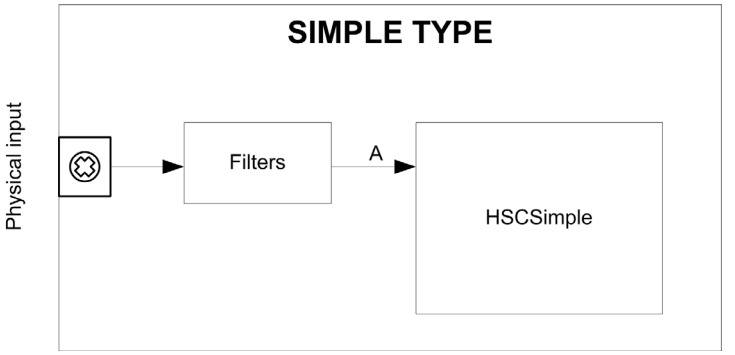
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A is the counting input of the `High Speed Counter`.

A **Simple** type can only count up. **Simple** type counting for **Modulo-loop** mode only counts down. **Simple** type counting for **One-shot** mode only counts up.

Configuration of the Simple Type in Modulo-loop Mode

Configuration Procedure

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

Step	Action
1	Double-click **Expert → DM72F• → HSCSimple•**.
2	Select the **HSC Simple Configuration** tab.
3	Set the value of the **General → Counting Mode** parameter to **Modulo-loop**.
4	Select the value of the **Counting inputs → A input → Bounce filter** parameter.
5	Enter the value of the **Range → Modulo** parameter.

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

Programmable Filter

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

Input	Filter value	Maximum counter frequency
A	0.002 ms	200 kHz
	0.004 ms (default value)	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

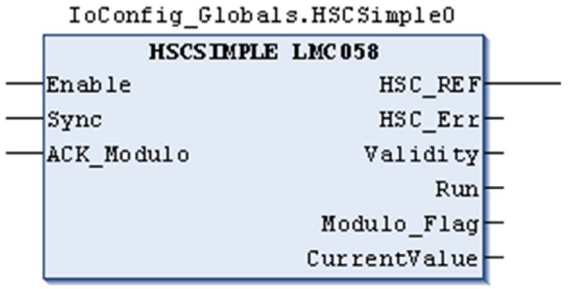
Overview

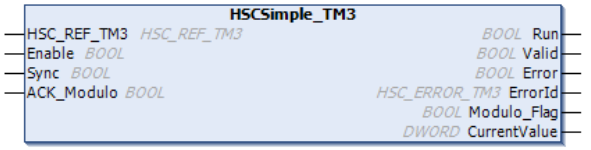
A **Simple** type is always managed by an HSCSimple function block.

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

Adding a HSCSimple Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → HSC → HSCSimple_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Simple** type instance name (defined in configuration) or select the function block instance by clicking: ??? ... 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.

This table describes the input variables:

Input	Type	Comment
Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
Sync	BOOL	On rising edge, presets and starts the counter.
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag.

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	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 226)* 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	Not relevant
Modulo_Flag	BOOL	Set to TRUE when the counter rolls over the Modulo value.
CurrentValue	DWORD	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 228)* or EXPERTSetParam *(see page 230)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 211)* that can be read or modified while the program is running:

Parameter	Description
EXPERT_MODULO	to get or set the modulo value of an HSC

Chapter 9
Modulo-loop with a Main Type

Overview

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

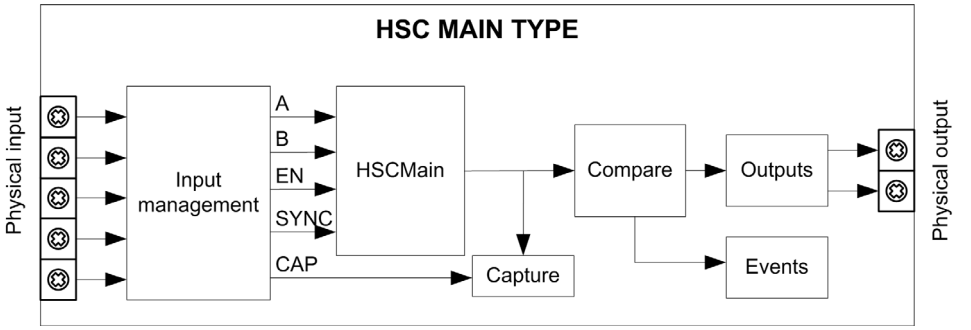
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A and B are the counting inputs of the counter.

EN is the enable input of the counter.

SYNC is the synchronization input of the counter.

CAP is the capture input of the counter.

Optional Functions

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

- Enable function *(see page 198)*
- Capture function *(see page 185)*
- Comparison function *(see page 175)*

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

Configuration of the Main Type in Modulo-loop Mode

Configuration Procedure

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

Step	Action
1	Double-click **Expert → DM72F• → HSCMain**.
2	Select the **HSC Main Configuration** tab.
3	Set the value of the **General → Counting Mode** parameter to **Modulo-loop**.
4	Select the value of the **General → Input Mode** parameter.
5	Select the value of the **Counting inputs → A input → Bounce filter** parameter. If necessary, select the value of the **Counting inputs → B input → Bounce filter** parameter.
6	Enter the value of the **Range → Modulo** parameter.
7	Optionnally, you can enable these functions: ● capture function *(see page 185)* ● comparison function *(see page 175)* ● enable function *(see page 198)* ● preset function *(see page 194)*

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

Programmable Filter

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

Input	Filter value	Maximum counter frequency
A, B	0.002 ms (default value)	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

Input Parameter

The value of the **Input mode** parameter determines the counting behavior:

Parameter Value	Description
A Single Phase	The counter increments on A.
A=UP, B=DOWN	The counter increments on A and decrements on B.
A=Pulse, B=Direction	• If there is a rising edge on A and B = True, the counter decrements. • If there is a rising edge on A and B = False, 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 • 1 count by encoder cycle
Normal Quadrature X2	• A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction • 2 counts by encoder cycle
Normal Quadrature X4	• A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction • 4 counts by encoder cycle
Reverse Quadrature X1	• A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction • 1 count by encoder cycle
Reverse Quadrature X2	• A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction • 2 counts by encoder cycle
Reverse Quadrature X4	• A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction • 4 counts by encoder cycle

Programming the Main Type

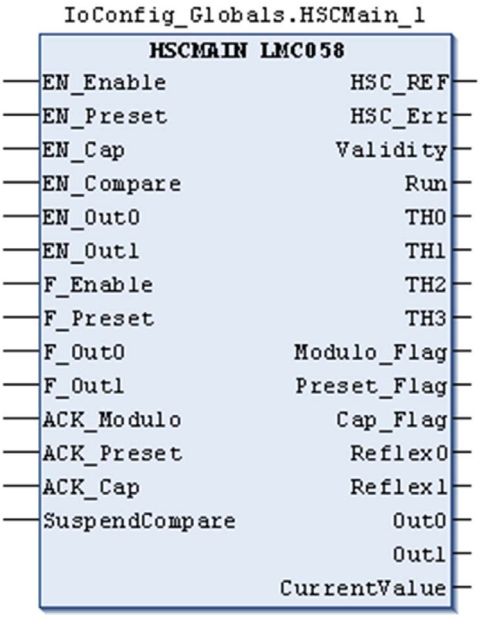
Overview

The **Main** type is always managed by an `HSCMain` function block.

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

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → HSC → HSCMain_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking: ??? ... 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.

This 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 *(see page 198)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter synchronization and start via the Sync input *(see page 194)*.
EN_Cap	BOOL	When **CAP** input is configured: if TRUE, enables the Capture input *(see page 186)*.
EN_Compare	BOOL	TRUE = enables the comparison function *(see page 175)* using Threshold 0, 1, 2, 3: • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1 output bits) • events (to trigger external tasks on threshold crossing)
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 *(see page 198)*. Takes priority over EN_Enable input.
F_Preset	BOOL	Forces the Preset condition *(see page 194)*. Takes priority over EN_Preset input.
F_Out0	BOOL	TRUE = forces Output0 to 1 (if Reflex0 is configured in EcoStruxure Machine Expert HSC Embedded Functions). Takes priority over EN_Out0.
F_Out1	BOOL	TRUE = forces Output1 to 1 (if Reflex1 is configured in EcoStruxure Machine Expert HSC Embedded Functions). Takes priority over EN_Out1.
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: • 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.

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	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 226)* 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 activated.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 *(see page 175)*.
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 *(see page 175)*.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 *(see page 175)*.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 *(see page 175)*.
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 194)*.
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 187)*. This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0 *(see page 176)*.
Reflex1	BOOL	State of Reflex1 *(see page 176)*.
Out0	BOOL	State of physical output Output0 to 1 (if Reflex0 configured in EcoStruxure Machine Expert HSC Embedded Functions, otherwise FALSE if not configured).
Out1	BOOL	State of physical output Output1 to 1 (if Reflex1 configured in EcoStruxure Machine Expert HSC Embedded Functions, otherwise FALSE if not configured).
CurrentValue	DINT	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 228)* or EXPERTSetParam *(see page 228)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 211)* that can be read or modified while the program is running:

Parameter	Description
EXPERT_MODULO	to get or set the Modulo value of an HSC
EXPERT_THRESHOLD0	to get or set the Threshold 0 value of an HSC
EXPERT_THRESHOLD1	to get or set the Threshold 1 value of an HSC
EXPERT_THRESHOLD2	to get or set the Threshold 2 value of an HSC
EXPERT_THRESHOLD3	to get or set the Threshold 3 value of an HSC
EXPERT_REFLEX0	to get or set output 0 reflex mode of an EXPERT function
EXPERT_REFLEX1	to get or set output 1 reflex mode of an EXPERT function

Part IV
Free-large Mode

Overview

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

What Is in This Part?

This part contains the following chapters:

Chapter 10
Free-large Mode Principle

Overview

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

What Is in This Chapter?

This chapter contains the following topics:

Free-large Mode Principle Description

Overview

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

Principle

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

When counting is enabled *(see page 198)*, 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 196)* which loads the preset value.

The current counter is stored in the capture register by using the Capture *(see page 185)* function.

If the counter reaches the counting limits, the counter will react according to the Limits Management *(see page 81)* configuration.

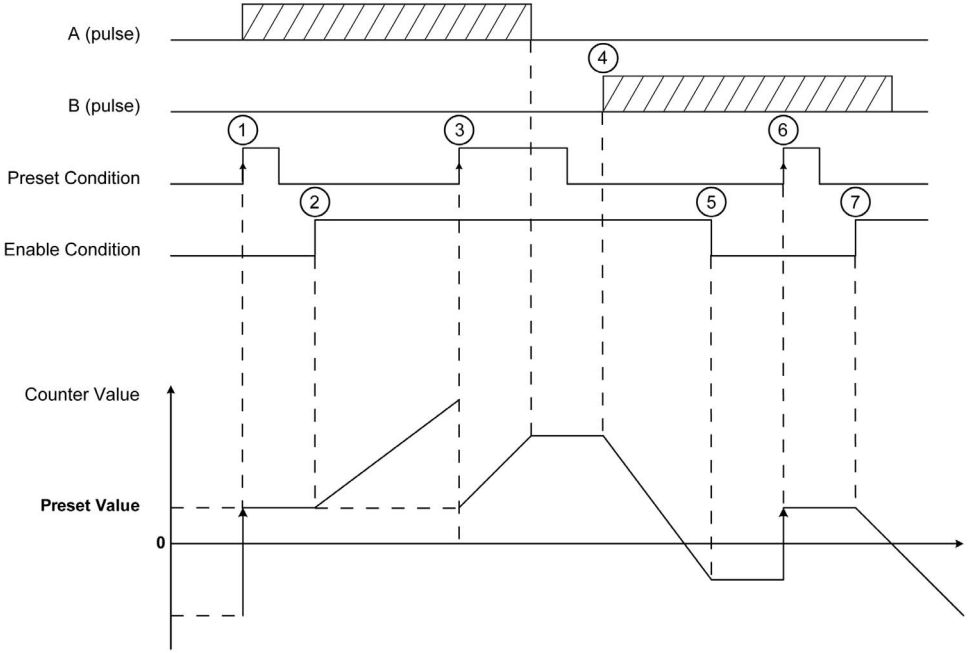
Input Modes

This table shows the 8 types of input modes available:

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

Up Down Principle Diagram

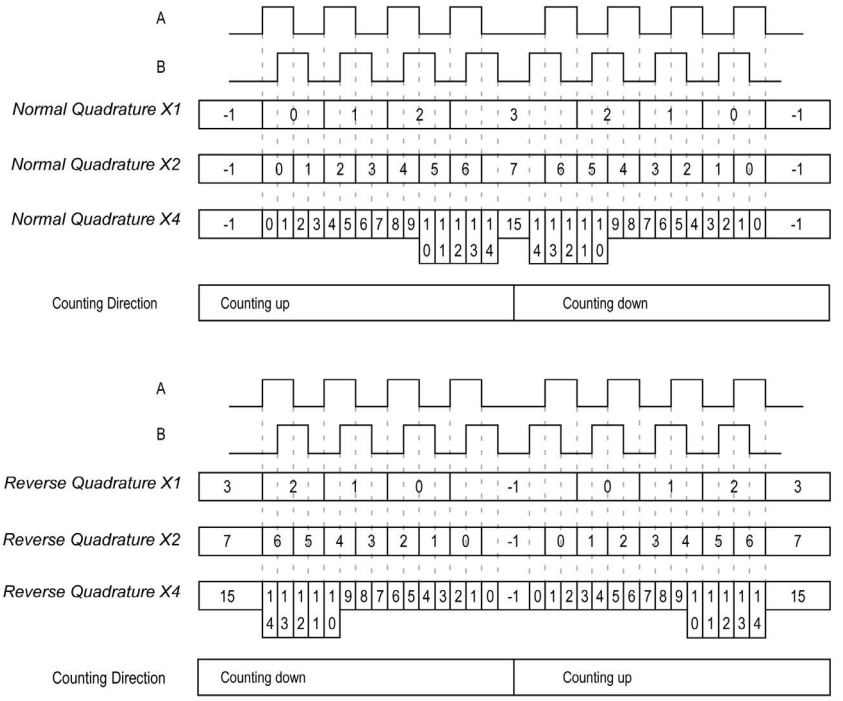
The figures shows the **A = Up, B = Down** mode:



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

Quadrature Principle Diagram

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



Limits Management

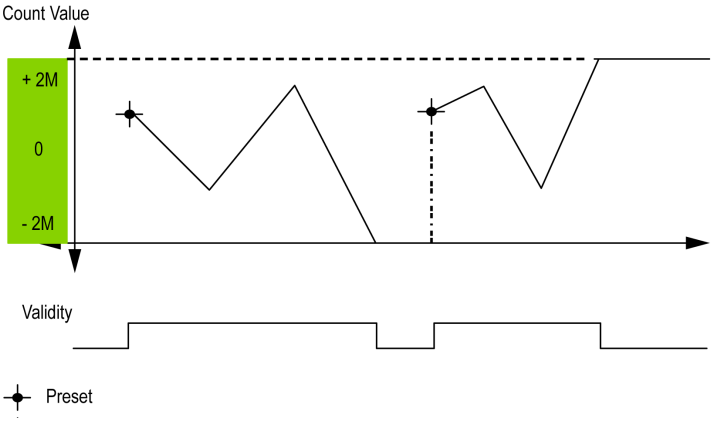
Overview

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

- Lock on limits
- Rollover

Lock on Limits

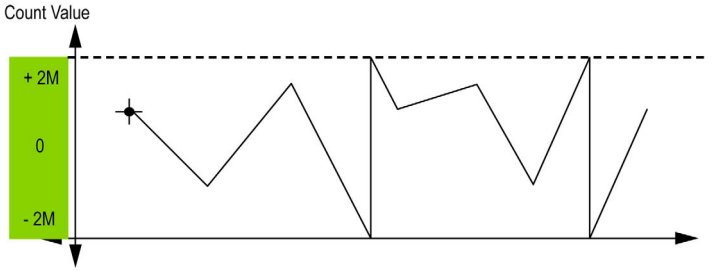
In the case of an overflow or underflow counter, the counter value is maintained at the limit value, the validity bit goes to 0, and the `Error` bit indicates that this detected error until the counter is preset again.



Rollover

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

`Modulo_Flag` (resp. Overflow_Flag for Encoder) output is set to 1.



Chapter 11
Free-large with a Main Type

Overview

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

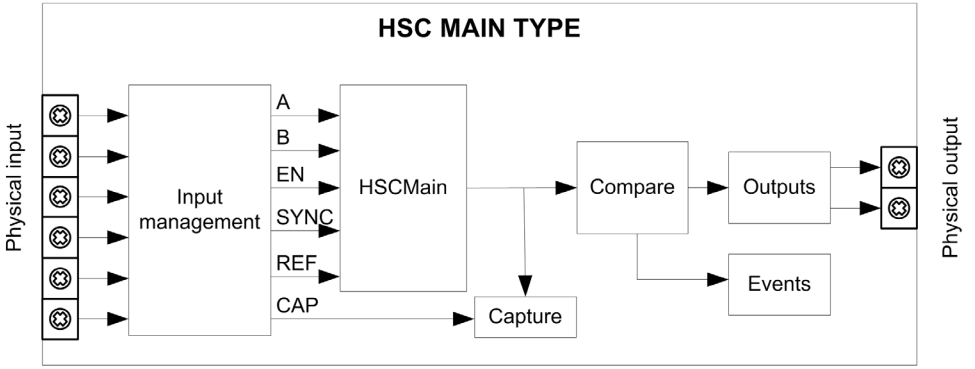
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A and B are the counting inputs of the counter.

EN is the enable input of the counter.

REF is the reference input of the counter.

SYNC is the synchronization input of the counter.

CAP is the capture input of the counter.

Optional Function

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

- Preset function *(see page 194)*
- Enable function *(see page 198)*
- Capture function *(see page 185)*
- Comparison function *(see page 175)*

Configuration of the Main Type in Free-large Mode

Configuration Procedure

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

Step	Action
1	Double-click **Expert → DM72F• → HSCMain**.
2	Select the **HSC Main Configuration** tab.
3	Set the value of the **General → Counting Mode** parameter to **Free-large**.
4	Select the value of the **General → Input Mode** parameter.
5	Select the value of the **Counting inputs → A input → Bounce filter** parameter. If necessary, select the value of the **Counting inputs → B input → Bounce filter** parameter.
6	Enter the value of the **Range → Preset** parameter. The **Limits** defines the behavior of the counter when limits are reached.
7	Optionnally, you can enable these functions: ● capture function *(see page 185)* ● comparison function *(see page 175)* ● enable function *(see page 198)* ● preset function *(see page 194)*

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

Programmable Filter

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

Input	Filter Value	Maximum Counter Frequency
A, B	0.002 ms (default value)	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

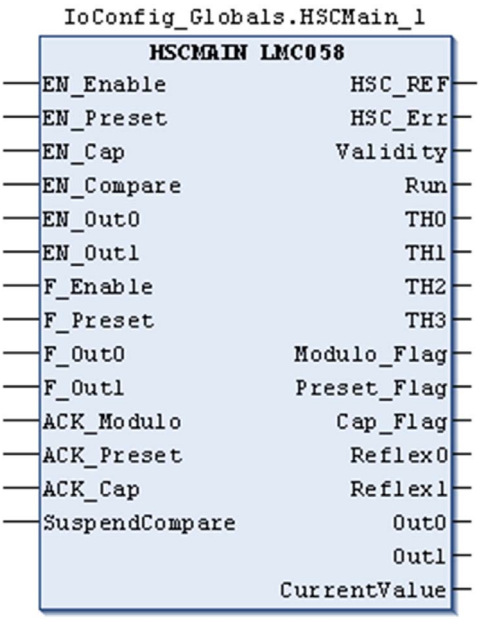
Overview

The **Main** type is always managed by an `HSCMain` function block.

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

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → HSC → HSCMain_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking:  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.

This 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 *(see page 198)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter synchronization and start via the Sync input *(see page 194)*.
EN_Cap	BOOL	When **CAP** input is configured: if TRUE, enables the Capture input *(see page 188)*.
EN_Compare	BOOL	TRUE = enables the comparison operation *(see page 175)* (using Thresholds 0, 1, 2, 3): • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1 output bits) • events (to trigger external tasks on threshold crossing)
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 *(see page 198)*. Takes priority over EN_Enable input.
F_Preset	BOOL	Forces the Preset condition *(see page 194)*. Takes priority over EN_Preset input.
F_Out0	BOOL	TRUE = forces Output0 to 1 (if Reflex0 is configured in HSC Embedded Functions). Takes priority over EN_Out0.
F_Out1	BOOL	TRUE = forces Output1 to 1 (if Reflex1 is configured in HSC Embedded Functions). Takes priority over EN_Out1.
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: • 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.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	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 226)* 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 used.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 *(see page 175)*.
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 *(see page 175)*.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 *(see page 175)*.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 *(see page 175)*.
Modulo_Flag	BOOL	Set to 1 when the counter rolls over its limits.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 194)*
Cap_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 186)*. This flag must be reset before a new capture can occur.
Reflex0	BOOL	State of Reflex0. *(see page 176)*
Reflex1	BOOL	State of Reflex1. *(see page 176)*
Out0	BOOL	State of physical outputs Output0 (if Reflex0 is configured in HSC Embedded Function, otherwise FALSE if not configured).
Out1	BOOL	State of physical outputs Output1 (if Reflex1 is configured in HSC Embedded Function, otherwise FALSE if not configured).
CurrentValue	DINT	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the EXPERTGetParam *(see page 228)* or EXPERTSetParam *(see page 230)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 211)* enumeration which can be read or modified while the program is running:

Parameter	Description
EXPERT_PRESET	to get or set the Preset value of the HSC
EXPERT_THRESHOLD0	to get or set the Threshold 0 value of an HSC
EXPERT_THRESHOLD1	to get or set the Threshold 1 value of an HSC
EXPERT_THRESHOLD2	to get or set the Threshold 2 value of an HSC
EXPERT_THRESHOLD3	to get or set the Threshold 3 value of an HSC
EXPERT_REFLEX0	to get or set output 0 reflex mode of an expert function
EXPERT_REFLEX1	to get or set output 0 reflex mode of an expert function

Part V
Event Counting Mode

Overview

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

What Is in This Part?

This part contains the following chapters:

Chapter 12
Event Counting Principle

Event Counting Mode Principle Description

Overview

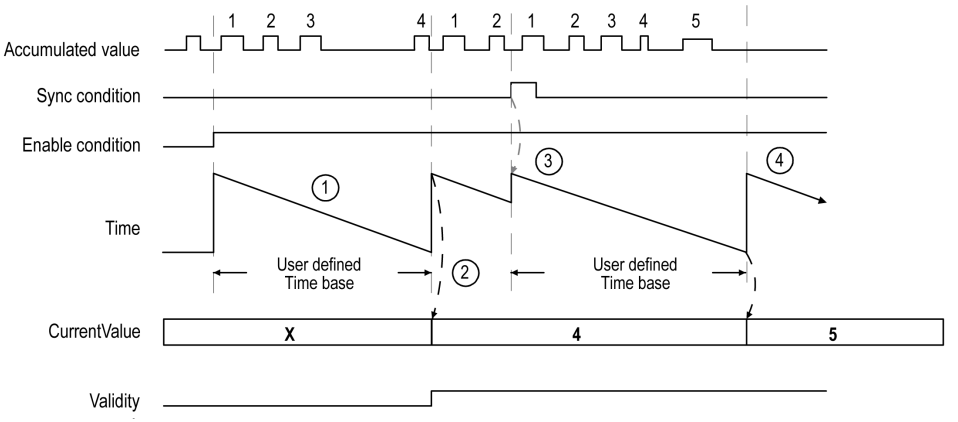
The **Event Counting** mode allows you to count the number of events that occur during a given period of time.

Principle

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

Synchronization can be used over the time period. This restarts the counting event for a new predefined time period. The counting restarts at the edge Sync condition *(see page 194)*.

Principle Diagram



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

Stage	Action
3	On the rising edge of the Sync condition: ● the accumulated value is reset to 0 ● the counter value is not updated ● the counting restarts for a new period of time
4	Once the period of time has elapsed, the counter value is set to the number of events counted over the period. The counting restarts for a new period of time.

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

Chapter 13
Event Counting with a Main Type

Overview

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

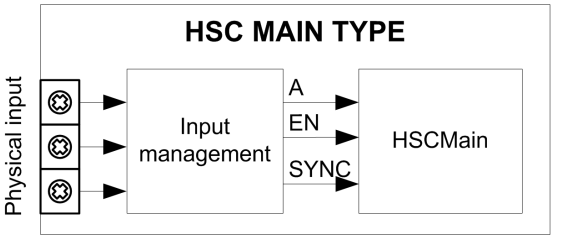
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A is the counting input of the counter.

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:

- Preset function *(see page 194)*
- Enable function *(see page 198)*

Configuration of the Main Type in Event Counting Mode

Configuration Procedure

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

Step	Action
1	Double-click **Expert → DM72F• → HSCMain**.
2	Select the **HSC Main Configuration** tab.
3	Set the value of the **General → Counting Mode** parameter to **Event**.
4	Select the value of the **Counting inputs → A input → Bounce filter** parameter.
5	Set the time base value from **Range → Time base**.
6	Optionally, select the value of the **SYNC** auxiliary input from the drop down to enable the Preset function *(see page 194)* with a physical input.

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

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 (default value)	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

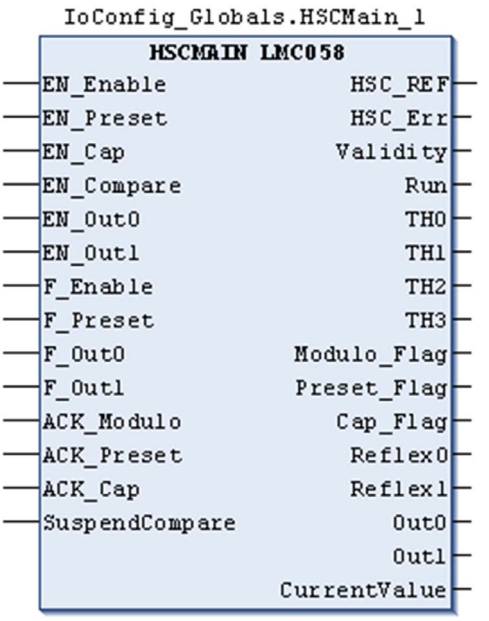
Overview

The **Main** type is always managed by an `HSCMain` function block.

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

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → HSC → HSCMain_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking: ??? ... 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

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

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	When **EN** input is configured: if TRUE, authorizes the counter enable via the Enable input *(see page 198)*.
EN_Preset	BOOL	When **SYNC** input is configured: if TRUE, authorizes the counter synchronization and start via the Sync input *(see page 194)*.
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 *(see page 198)*. Takes priority over EN_Enable input.
F_Preset	BOOL	Forces the Preset condition *(see page 194)*. Takes priority over EN_Preset input.
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.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	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 226)* 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 used.
TH0	BOOL	Not used.
TH1	BOOL	Not used.
TH2	BOOL	Not used.

Outputs	Type	Comment
TH3	BOOL	Not used.
Modulo_Flag	BOOL	Not used.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 194)*.
Cap_Flag	BOOL	Not used.
Reflex0	BOOL	Not used.
Reflex1	BOOL	Not used.
Out0	BOOL	Not used.
Out1	BOOL	Not used.
CurrentValue	DINT	Current value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the `EXPERTGetParam` *(see page 228)* or `EXPERTSetParam` *(see page 230)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start.

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` *(see page 211)* which can be read or modified while the program is running:

Parameter	Type	Description
EXPERT_TIMEBASE	`EXPERT_TIMEBASE_TYPE` For more information, refer to Type for HSC *(see page 213)*.	To get or set the Timebase value of the HSC.
EXPERT_PERIODMETER_RESOLUTION_TYPE	`EXPERT_PERIODMETER_RESOLUTION_TYPE` For more information, refer to Type for HSC *(see page 210)*.	To dynamically read or modify the time base.

Part VI
Frequency Meter Type

Overview

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

What Is in This Part?

This part contains the following chapters:

Chapter 14
Frequency Meter Principle

Description

Overview

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

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

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

Operation Limits

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

The maximum duty cycle at 200 kHz is 60%.

Chapter 15
Frequency Meter with a Main Type

Overview

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

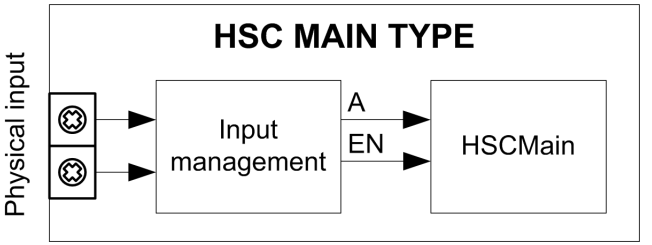
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A is the counting input of the counter.

EN is the enable input of the counter.

Optional Function

In addition to the **Frequency meter** type, the **Main** type can provide the Enable function *(see page 198)*.

Configuration

Configuration Procedure

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

Step	Action
1	Double-click **Expert** → **DM72F•** → **HSCMain**.
2	Select the **HSC Main Configuration** tab.
3	Set the value of the **General** → **Counting Mode** parameter to **Frequency meter**.
4	Select the value of the **Counting inputs** → **A input** → **Bounce filter** parameter.
5	Set the time base value from **Range** → **Time base**.
6	Optionally, select the value of the **EN** auxiliary input from the drop down to enable the Enable function *(see page 198)* with a physical input.

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

Programmable Filter

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

Input	Filter Value	Maximum Counter Frequency
A	0.002 ms (default value)	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

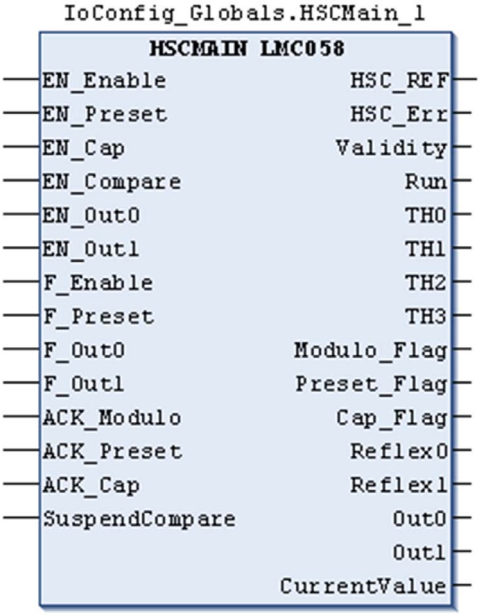
Overview

The **Main** type is always managed by an `HSCMain` function block.

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

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → HSC → HSCMain_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking:  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** type.

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	If TRUE and the **EN** input is configured, authorizes the counter to be enabled using the Enable input *(see page 198)*.
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 *(see page 198)*.
F_Preset	BOOL	Forces the Preset condition *(see page 194)*
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.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	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 226)* 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 used.
TH0	BOOL	Not used.
TH1	BOOL	Not used.
TH2	BOOL	Not used.
TH3	BOOL	Not used.

Outputs	Type	Comment
Modulo_Flag	BOOL	Not used.
Preset_Flag	BOOL	Set to 1 by the preset of the counter *(see page 194)*
Cap_Flag	BOOL	Not used.
Reflex0	BOOL	Not used.
Reflex1	BOOL	Not used.
Out0	BOOL	Not used.
Out1	BOOL	Not used.
CurrentValue	DINT	The value of the counter.

Adjusting Parameters

Overview

The list of parameters described in the table can be read or modified by using the `EXPERTGetParam` *(see page 228)* or `EXPERTSetParam` *(see page 230)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start.

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_FREQMETER_PARAMETER_TYPE` *(see page 207)* which can be read or modified while the program is running:

Parameter	Type	Description
EXPERT_FREQMETER_PARAMETER_TYPE	EXPERT_FREQMETER_PARAMETER_TYPE For more information, refer to Type for HSC *(see page 207)*.	To get or set the Expert functions.

Part VII
Period Meter Type

Overview

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

What Is in This Part?

This part contains the following chapters:

Chapter 16
Period Meter Type Principle

Description

Overview

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

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

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

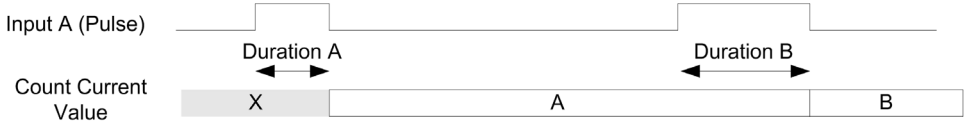
For example, if the counter value = 100 and the **Resolution** parameter is:
0.0001 (0.1 µs) measurement = 0.01 ms
0.001 (1 µs) measurement = 0.1 ms
0.1 (100 µs) measurement = 10 ms
1 (1000 µs) measurement = 100 ms

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

Edge to Opposite Mode

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

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



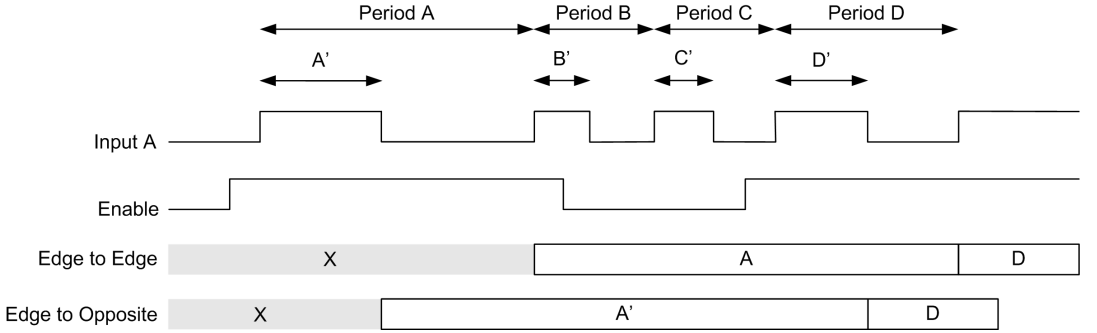
Edge to Edge Mode

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

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

Enable Condition Interruption Behavior

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



Operating Limits

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

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

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

Chapter 17
Period Meter with a Main Type

Overview

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

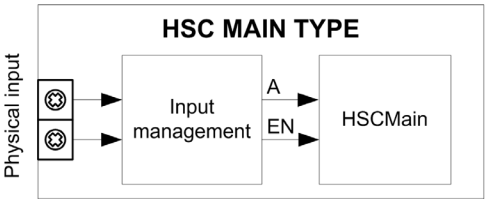
What Is in This Chapter?

This chapter contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A is the counting input of the counter.

EN is the enable input of the counter.

Optional Function

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

- Enable function *(see page 198)*

Configuration

Configuration Procedure

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

Step	Action
1	Double-click **Expert → DM72F• → HSCMain**.
2	Select the **HSC Main Configuration** tab.
3	Set the value of the **General → Counting Mode** parameter to **Period meter**.
4	Set the period meter mode from **General → PeriodMeter Mode**.
5	Select the value of the **Counting inputs → A input → Bounce filter** parameter.
6	Optionally, select the value of the **EN** auxiliary input from the drop down to enable the Enable function *(see page 198)* with a physical input.
7	Set the resolution value from **Range → Resolution**.
8	Set the time-out value from **Range → Timeout**.

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

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 (default value)	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

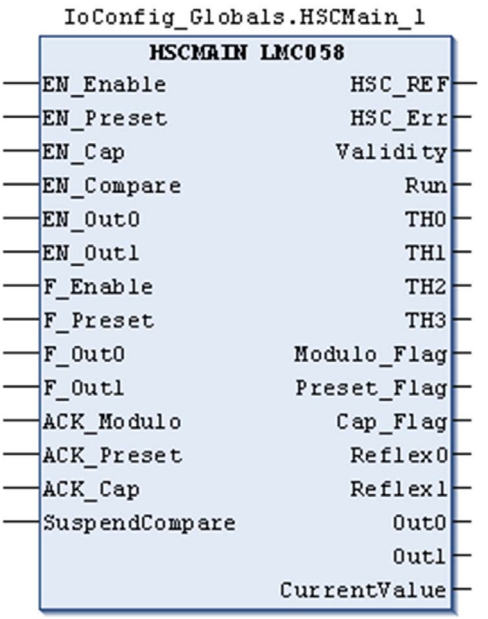
Overview

The **Main** type is always managed by an `HSCMain` function block.

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

Adding the HSCMain Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → HSC → HSCMain_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the **Main** type instance name (defined in configuration) or select the function block instance by clicking:  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** type.

This 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 *(see page 198)*.
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 *(see page 198)*.
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.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	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 226)* 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 activated.
TH0	BOOL	Not used.
TH1	BOOL	Not used.
TH2	BOOL	Not used.
TH3	BOOL	Not used.

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

Adjusting Parameters

Overview

The list of parameters described in the table below can be read or modified by using the EXPERTGetParam *(see page 228)* or EXPERTSetParam *(see page 230)* function blocks.

NOTE: Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on a cold or warm start.

Adjustable Parameters

This table provides the list of parameters from the EXPERT_PARAMETER_TYPE *(see page 211)* which can be read or modified while the program is running:

Parameter	Description
EXPERT_TIMEBASE	To get or set the Resolution value of the HSC.
EXPERT_PERIODMETER_RESOLUTION_TYPE	To dynamically read or modify the time base. For more information, refer to Type for period meter *(see page 210)*.

Part VIII
Encoder

Overview

This part describes how to implement an encoder.

What Is in This Part?

This part contains the following chapters:

Chapter 18
Incremental Mode with an Encoder

Overview

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

What Is in This Chapter?

This chapter contains the following sections:

Section 18.1
Incremental Mode Principle

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 198)* 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 194)* occurs.

The current counter can be stored in the capture register by configuring the Capture conditions *(see page 189)*.

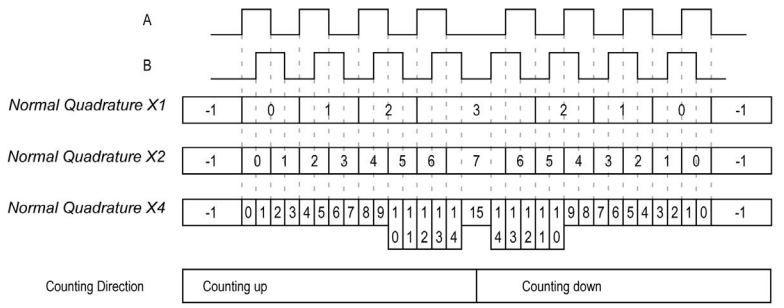
Axis Types

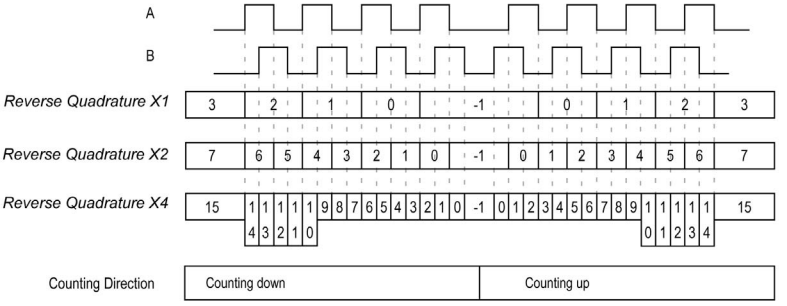
The following table shows the 2 available axis types:

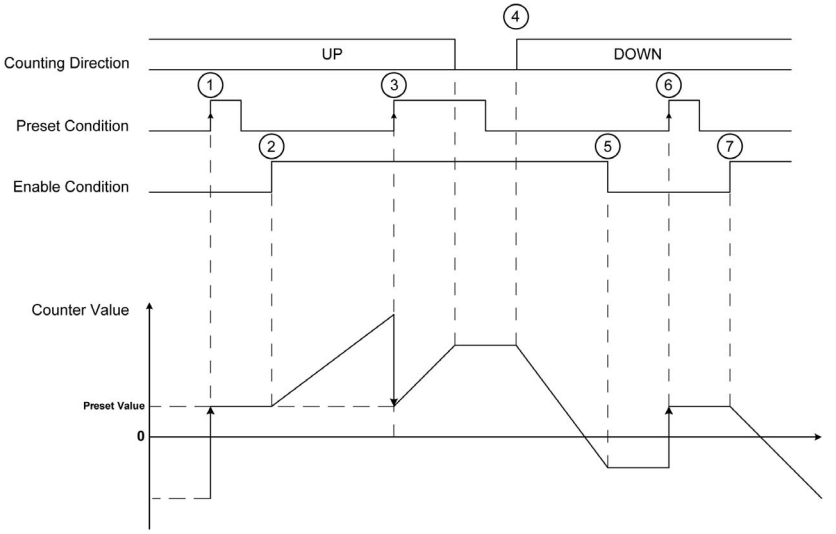
Axis Type	Comment
Linear	This mode acts as a finite counter.
Rotary	This mode acts as an infinite counter.

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.

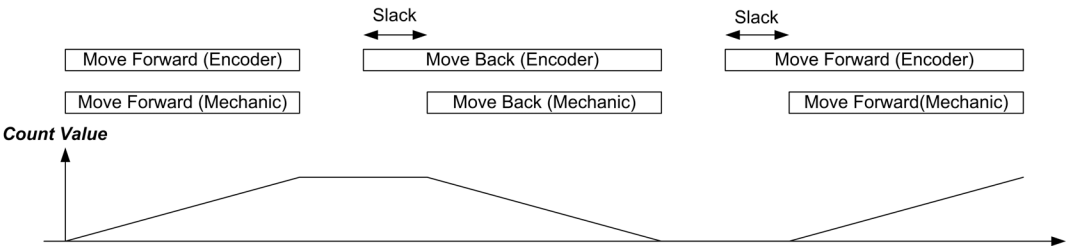
NOTE: Enable and Preset conditions depends on configuration. These are described in the Enable *(see page 198)* and Preset *(see page 196)* function.

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:



Section 18.2
Standard Encoder on an Expert I/O Module

Overview

This section describes the configuration in **Incremental Mode** of a **Standard Encoder** on an **Expert I/O** module.

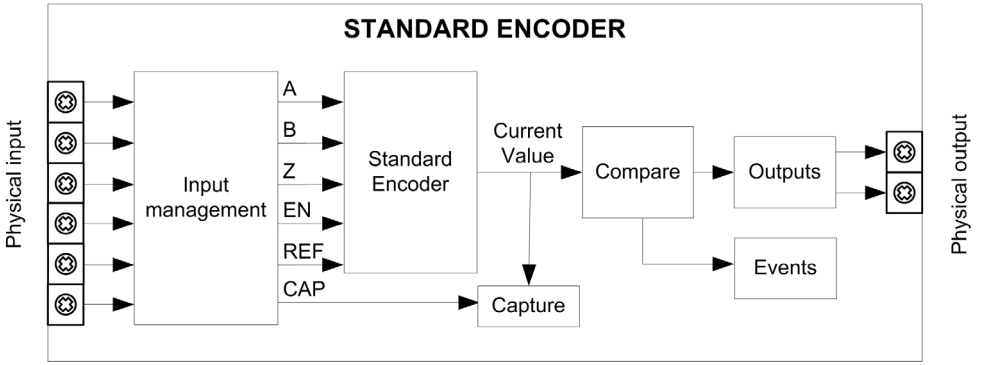
What Is in This Section?

This section contains the following topics:

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 functions:

- Compare *(see page 175)*
- Capture *(see page 189)*
- Enable with a physical input *(see page 198)*
- Preset a physical input *(see page 194)*

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

Configuration Window

Follow this procedure to configure the **Standard Encoder** type in **Incremental** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → Expert → DM72Fx → Standard_Encoder.** **Result**: The configuration window is displayed.
2	Select the **Standard Encoder Configuration** tab.

I/O Configuration Parameters

This table describes the properties of the **Standard Encoder Configuration** tab in **Incremental** mode:

Parameter			Value	Default Value	Description
General	**Input Mode**		Normal Quadrature X1 Normal Quadrature X2 Normal Quadrature X4 Reverse Quadrature X1 Reverse Quadrature X2 Reverse Quadrature X4	Normal Quadrature X1	Select the period measurement interval.
Counting inputs	**A input**	**Bounce filter**	0.002 0.004 0.012 0.04 0.12 0.4 1.2 4	0.002	Set the filtering value (in ms) to reduce the bounce effect on the A input.
Range	**Preset**		0	0	Set the counting initial value.
	Limits		Lock on limits Modulo	Lock on limits	Limits management.
	Slack correction		0	0	Specify the slack correction value.

Parameter			Value	Default Value	Description
Control inputs	**Z input**	**Location**	Disabled I1.2	Disabled	Select the Z input used for presetting the counting function.
		Bounce filter	0.002 0.004 0.012 0.04 0.12 0.4 1.2 4	0.002	Set the filtering value (in ms) to reduce the bounce effect on the Z input.
	EN input	**Location**	Disabled I1.4	Disabled	Select the PLC input used for enabling the counting function.
	REF input	**Location**	Disabled I1.5	Disabled	Select the REF input used for presetting the counting function.
	Edge detection		None Z Rising Z Falling Z Both	None	Select the condition to preset the counting function with the SYNC and REF inputs.
Capture	**CAP input**	**Location**	Disabled I1.3	Disabled	Select the PLC input used for capturing the current counting value.
	CAP0 Mode		None Z Rising	None	Select the condition to perform a first capture of the counting current value.
	CAP1 Mode		None Z Rising	None	Select the condition to perform a second capture of the counting current value.
Compare	**Thresholds**	**Number of thresholds**	0 1 2 3 4	0	Select the number of thresholds (0...4).
Scaling	**Axis Type**		Linear Rotary	Linear	Select the axis type.
	Increments		2048	2048	Increment Per Turn
	Units		360	360	Unit Per Turn

⚠ WARNING

INACCURATE ENCODER VALUE

Respect the mathematical rule of modulo ((**Modulo x Increment)/Unit** = Integer) to avoid slipping.

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

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

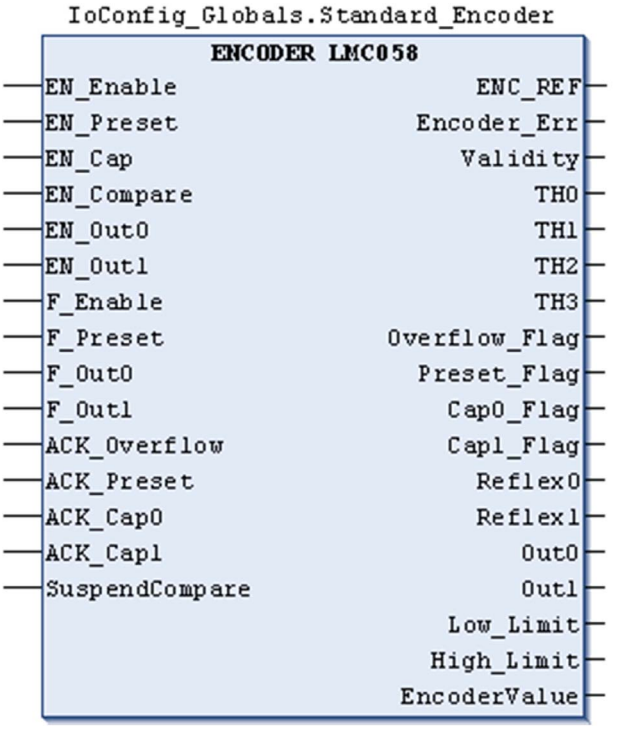
Programming the Standard Encoder

Overview

A **Standard Encoder** is always managed by an Encoder_LMC058 *(see page 232)* function block.

Adding a StandardEncoder Function Block

Step	Action
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → ENCODER → ENCODER_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the `Encoder_LMC058` instance name or select the function block instance by clicking on: `???` `...` Using the input assistant, the `Encoder_LMC058` 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 *(see page 198)*.
EN_Preset	BOOL	When **Z** or **REF** input are configured authorizes the counter preset via the inputs *(see page 194)*.
EN_Cap	BOOL	When at least one **CAP** input is configured authorizes the capture function via the inputs *(see page 189)*.
EN_Compare	BOOL	TRUE = enables the comparator operation using Thresholds 0, 1, 2, 3 *(see page 175)*: • 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 *(see page 198)*.
F_Preset	BOOL	Forces the Preset condition.
F_Out0	BOOL	TRUE = forces physical output Output0 to 1 (if Reflex0 is configured) *(see page 176)*.
F_Out1	BOOL	TRUE = forces physical output Output1 to 1 (if Reflex1 is configured) *(see page 176)*.
ACK_Overflow	BOOL	On rising edge, resets Overflow_Flag
ACK_Preset	BOOL	On rising edge, resets Preset_Flag *(see page 194)*.
ACK_Cap0	BOOL	On rising edge, resets Cap0_Flag *(see page 189)*.
ACK_Cap1	BOOL	On rising edge, resets Cap1_Flag *(see page 189)*.
SuspendCompare	BOOL	TRUE = the comparator operation results are frozen *(see page 175)*: • TH0, TH1, TH2, TH3, Reflex0, Reflex1 output bits maintain their last value. • Physical outputs Output0 and Output1 maintain their last value. • Events are masked. EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SupendCompare is set.

The following table describes the output variables:

Outputs	Type	Comment
ENC_REF	EXPERT_REF *(see page 212)*	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 226)* 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 176)*.
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 (if configured) *(see page 176)*.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (if configured) *(see page 176)*.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (if configured) *(see page 176)*.
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 196)*.
Cap0_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 189)*. 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 189)*. This flag must be reset before a new capture is allowed.
Reflex0	BOOL	State of Reflex0 *(see page 176)*.
Reflex1	BOOL	State of Reflex1 *(see page 176)*.
Out0	BOOL	State of Output0 *(see page 176)*.
Out1	BOOL	State of Output1 *(see page 176)*.
Low_Limit	BOOL	Set to 1 when the encoder exceeds -2.147.483.648. *(see page 81)* Reset to 0 when encoder presets.
High_Limit	BOOL	Set to 1 when the encoder exceeds +2.147.483.647. *(see page 81)* 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 228)* or `EXPERTSetParam` *(see page 230)* function blocks.

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` *(see page 211)* which can be modified while the program is running:

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

Section 18.3
Standard Encoder on the Encoder Interface

Overview

This section describes the configuration in **Incremental Mode** of the **Standard Encoder** on the **Encoder** interface.

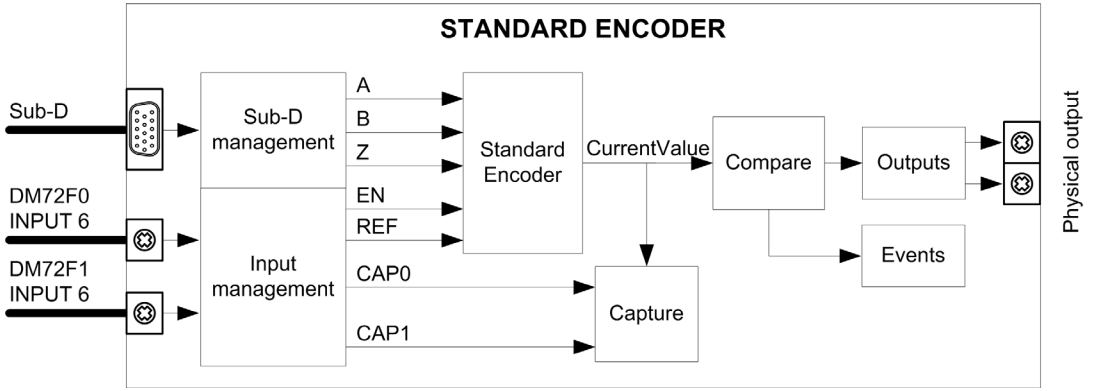
What Is in This Section?

This section contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A and B are the counting inputs of the counter.

EN is the enable input of the counter.

Z and REF are the reference inputs of the counter.

CAP is the capture input of the counter.

Optional Function

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

- Compare *(see page 175)*
- Capture *(see page 189)*
- Enable with a physical input *(see page 198)*
- Preset with physical inputs *(see page 196)*

Configuration of the Standard Encoder on the Encoder Interface

Configuration Window

Follow this procedure to configure the **Standard Encoder** type in **Incremental** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → Expert → Encoder → Standard_Encoder_1.** **Result**: The configuration window is displayed.
2	Select the **Standard Encoder Configuration** tab.

Configuration Parameters

This table describes the properties of the **Standard Encoder Configuration** tab in **Incremental** mode:

Parameter			Value	Default Value	Description
General	**Mode**		Incremental SSI	Incremental	Select the encoding mode.
	Input Mode		Normal Quadrature X1 Normal Quadrature X2 Normal Quadrature X4 Reverse Quadrature X1 Reverse Quadrature X2 Reverse Quadrature X4	Normal Quadrature X1	Select the period measurement interval.
Counting inputs	**A input**	**Bounce filter**	0.002 0.004 0.012 0.04 0.12 0.4 1.2 4	0.002	Set the filtering value (in ms) to reduce the bounce effect on the A input.
Range	**Preset**		0	0	Set the counting initial value.
	Limits		Lock on limits Modulo	Lock on limits	Limits management
	Slack correction		0	0	Specify the slack correction value.

Parameter			Value	Default Value	Description
Control inputs	**Z input**	**Bounce filter**	0.002 0.004 0.012 0.04 0.12 0.4 1.2 4	0.002	Set the filtering value (in ms) to reduce the bounce effect on the Z input.
	EN input	**Location**	Disabled I1.6	Disabled	Select the PLC input used for enabling the counting function.
	REF input	**Location**	Disabled I3.6	Disabled	Select the REF input used for presetting the counting function.
	Edge detection		None Z Rising Z Falling Z Both	None	Select the condition to preset the counting function with the SYNC and REF inputs.
Capture	**CAP 0 input**	**Location**	Disabled I1.6	Disabled	Select the PLC input used for capturing the first counting value.
	CAP 1 input	**Location**	Disabled I3.6	Disabled	Select the PLC input used for capturing the second counting value.
	CAP0 Mode		None Z Rising	None	Select the condition to perform a first capture of the counting current value.
	CAP1 Mode		None Z Rising	None	Select the condition to perform a second capture of the counting current value.
Compare	**Thresholds**	**Number of thresholds**	0 1 2 3 4	0	Select the number of thresholds (0...4).
Scaling	**Axis Type**		Linear Rotary	Linear	Select the axis type.
	Increments		2048	2048	Increment Per Turn
	Units		360	360	Unit Per Turn
Monitoring	**Power supply monitor**		Disabled Enabled	Disabled	Enable the power supply monitor.

⚠ WARNING
INACCURATE ENCODER VALUE Respect the mathematical rule of modulo ((**Modulo x Increment)/Unit** = Integer) to avoid slipping. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

Programmable Filter

The filtering value on the **Standard Encoder** input influences 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

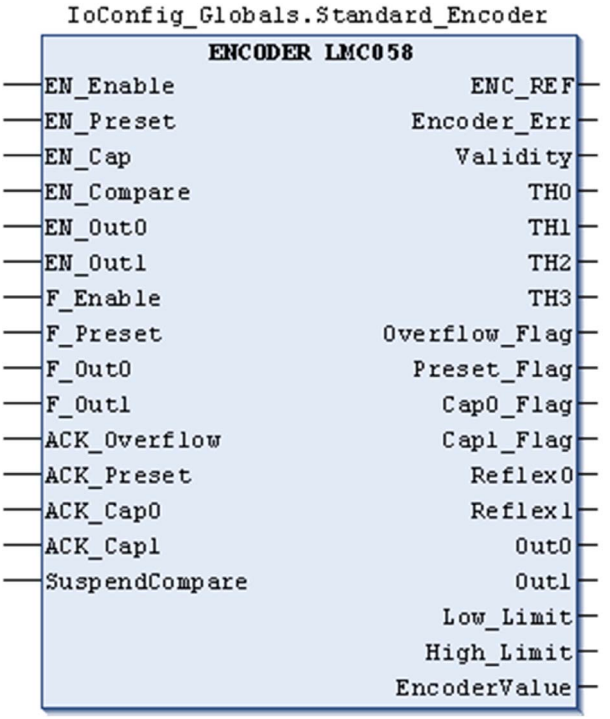
Programming the Standard Encoder

Overview

A **Standard Encoder** is always managed by an Encoder_LMC058 *(see page 232)* function block.

Adding a Standard Encoder Function Block

Step	Action
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → ENCODER → ENCODER_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the `Encoder_LMC058` instance name or select the function block instance by clicking: `???` `...` Using the input assistant, the `Encoder_LMC058` 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 *(see page 198)*.
EN_Preset	BOOL	When **Z** or **REF** input are configured authorizes the counter preset via the inputs *(see page 194)*.
EN_Cap	BOOL	When at least one **CAP** input is configured authorizes the capture function via the inputs *(see page 189)*.
EN_Compare	BOOL	TRUE = enables the comparator operation using Thresholds 0, 1, 2, 3 *(see page 175)*: • 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 0 to echo the Reflex0 value.
EN_Out1	BOOL	TRUE = authorizes physical output 1 to echo the Reflex1 value.
F_Enable	BOOL	Forces the Enable condition *(see page 198)*.
F_Preset	BOOL	Forces the Preset condition.
F_Out0	BOOL	TRUE = forces physical output 0 to 1 (if Reflex0 is configured) *(see page 176)*.
F_Out1	BOOL	TRUE=forces physical output 1 to 1 (if Reflex1 is configured) *(see page 176)*.
ACK_Overflow	BOOL	On rising edge, resets Overflow_Flag
ACK_Preset	BOOL	On rising edge, resets Preset_Flag *(see page 194)*.
ACK_Cap0	BOOL	On rising edge, resets Cap0_Flag *(see page 189)*.
ACK_Cap1	BOOL	On rising edge, resets Cap1_Flag *(see page 189)*.
SuspendCompare	BOOL	TRUE = the comparator operation results are frozen *(see page 175)*: • TH0, TH1, TH2, TH3, Reflex0, Reflex1 output bits maintain their last value. • Physical Outputs 0, 1 maintain their last value. • Events are masked. EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SupendCompare is set.

The following table describes the output variables:

Outputs	Type	Comment
ENC_REF	EXPERT_REF *(see page 212)*	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 226)* 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 176)*.
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 (if configured) *(see page 176)*.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (if configured) *(see page 176)*.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (if configured) *(see page 176)*.
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 196)*.
Cap0_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 189)*. 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 189)*. This flag must be reset before a new capture is allowed.
Reflex0	BOOL	State of Reflex0 *(see page 176)*.
Reflex1	BOOL	State of Reflex1 *(see page 176)*.
Out0	BOOL	State of Output0 *(see page 176)*.
Out1	BOOL	State of Output1 *(see page 176)*.
Low_Limit	BOOL	Set to 1 when the encoder exceeds - 2.147.483.648. *(see page 81)* Resets to 0 when encoder presets.
High_Limit	BOOL	Set to 1 when the encoder exceeds + 2.147.483.647. *(see page 81)* Resets 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 228)* or `EXPERTSetParam` *(see page 230)* function blocks.

Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` *(see page 211)* which can be modified while the program is running:

Parameter	Description
`EXPERT_PRESET`	to get or set the Preset value of the Encoder
`EXPERT_THRESHOLD0`	to get or set the Threshold 0 value of an Encoder
`EXPERT_THRESHOLD1`	to get or set the Threshold 1 value of an Encoder
`EXPERT_THRESHOLD2`	to get or set the Threshold 2 value of an Encoder
`EXPERT_THRESHOLD3`	to get or set the Threshold 3 value of an Encoder
`EXPERT_OFFSET`	to get or set OFFSET of an Encoder
`EXPERT_SLACK`	to get or set Slack of an Encoder
`EXPERT_TIMEBASE`	to get or set timebase of an Encoder
`EXPERT_SCALING`	to get or set the Scaling parameter of an Encoder The scaling parameter (`ParamValue` of the function block) is made of 2 sub-parameters: **High significant INT** = Increments **Low significant INT** = Units
EXPERT_REFLEX0	to get or set output 0 reflex mode of an EXPERT function
EXPERT_REFLEX1	to get or set output 0 reflex mode of an EXPERT function

Section 18.4
Motion Encoder on an Expert I/O Module

Overview

This section describes the configuration in **Incremental Mode** of a **Motion Encoder** on an **Expert I/O** module.

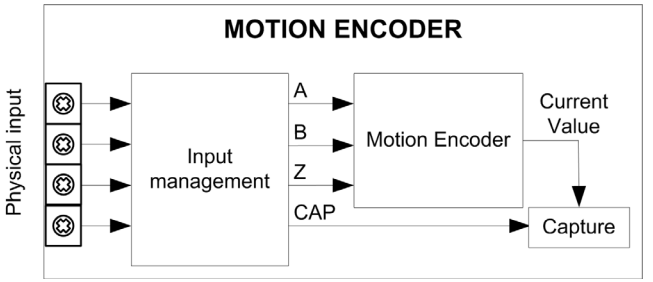
What Is in This Section?

This section contains the following topics:

Synopsis Diagram

Synopsis Diagram

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



A and B are the counting inputs of the encoder.

Z is the reference input of the encoder.

CAP is the capture input of the encoder.

Optional Function

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

- Capture *(see page 189)*

A **Motion Encoder** supports all SM3_Basic library SoftMotion functions (preset, touch prob, etc...).

In addition, an MC_GetImmediateValue_LMC058 *(see page 217)* function is available in the **LMC058 Motion** library.

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

Configuration Window

To configure the **Motion Encoder** type in **Incremental** mode, double-click **MyController → Expert → DM72Fx** in the **Devices tree**.

Result: The **Motion Encoder Configuration** window is displayed.

I/O Configuration Parameters

This table describes the properties of the **Motion Encoder Configuration** tab in **Incremental** mode:

Parameter			Value	Default Value	Description
General	**Input Mode**		Normal Quadrature X1 Normal Quadrature X2 Normal Quadrature X4 Reverse Quadrature X1 Reverse Quadrature X2 Reverse Quadrature X4	Normal Quadrature X1	Select the period measurement interval.
Counting inputs	**A input**	**Bounce filter**	0.002 0.004 0.012 0.04 0.12 0.4 1.2 4	0.002	Set the filtering value (in ms) to reduce the bounce effect on the A input.
Control inputs	**Z input**	**Location**	Disabled I3.2	Disabled	Select the Z input used for presetting the counting function.
Capture	**CAP input**	**Location**	Disabled I3.3	Disabled	Select the PLC input used for capturing the current counting value.

Programmable Filter

The filtering value on the **Motion 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

Section 18.5
Motion Encoder on the Encoder Interface

Overview

This section describes the configuration in **Incremental Mode** of a **Motion Encoder** on the **Encoder** interface.

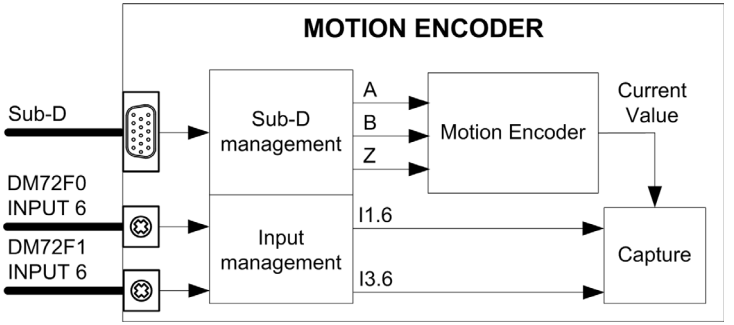
What Is in This Section?

This section contains the following topics:

Synopsis Diagram

Synopsis Diagram

The following diagram provides an overview of the **Motion Encoder** in **Incremental** mode on the **Encoder** interface:



A and B are the counting inputs of the encoder.

Z is the reference input of the encoder.

CAP0 and CAP1 are the capture inputs of the encoder.

Optional Function

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

- Capture *(see page 189)*

A **Motion Encoder** supports all SM3_Basic library SoftMotion functions (preset, touch prob, etc...).

In addition, an MC_GetImmediateValue_LMC058 *(see page 217)* function is available in the **LMC058 Motion** library.

Configuration of the Motion Encoder on the Encoder Interface

Configuration Window

To configure the **Motion Encoder** type in **Incremental** mode, double-click **MyController → Expert → Encoder → Motion_Encoder_1**.

Result: the **Motion Encoder Configuration** window is displayed.

Configuration Parameters

This table describes the properties of the **Motion Encoder Configuration** tab in **Incremental** mode:

Parameter			Value	Default Value	Description
General	**Mode**		Incremental SSI	Incremental	Select the encoding mode.
	Input Mode		Normal Quadrature X1 Normal Quadrature X2 Normal Quadrature X4 Reverse Quadrature X1 Reverse Quadrature X2 Reverse Quadrature X4	Normal Quadrature X1	Select the period measurement interval.
Counting inputs	**A input**	**Bounce filter**	0.002 0.004 0.012 0.04 0.12 0.4 1.2 4	0.002	Set the filtering value (in ms) to reduce the bounce effect on the A input.
Control inputs	**Z input**	**Bounce filter**	0.002 0.004 0.012 0.04 0.12 0.4 1.2 4	0.002	Set the filtering value (in ms) to reduce the bounce effect on the Z input.

Parameter			Value	Default Value	Description
Capture	**CAP 0 input**	**Location**	Disabled I1.6	Disabled	Select the PLC input used for capturing the first counting value.
	CAP 1 input	**Location**	Disabled I3.6	Disabled	Select the PLC input used for capturing the second counting value.
	CAP0 Mode		None Z Rising	None	Select the condition to perform a first capture of the counting current value.
	CAP1 Mode		None Z Rising	None	Select the condition to perform a second capture of the counting current value.
Monitoring	**Power supply monitor**		Disabled Enabled	Disabled	Enable the power supply monitor.

Programmable Filter

The filtering value on the **Motion Encoder** input determines 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

Chapter 19
SSI Mode with an Encoder

Overview

This chapter describes each of the encoders available in **SSI** Mode.

What Is in This Chapter?

This chapter contains the following sections:

Section 19.1
SSI Principle Description

SSI Mode Principle Description

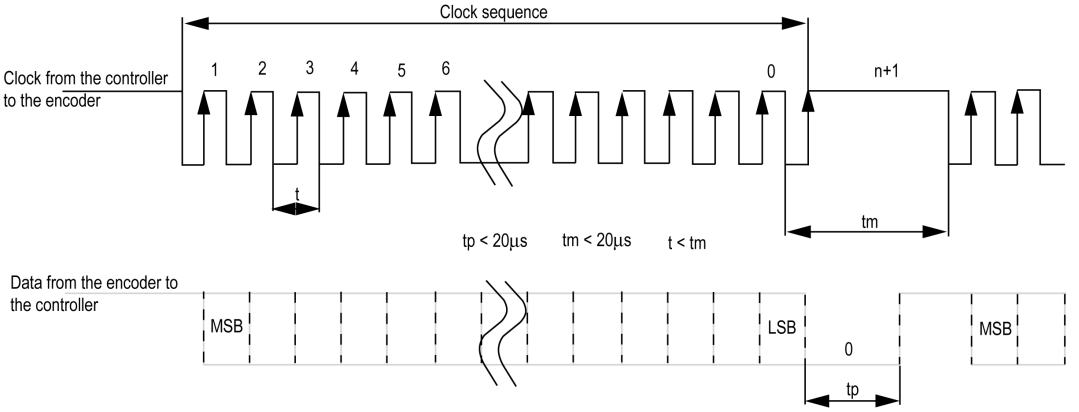
General

The SSI (Synchronous Serial Interface) mode allows the connection of an absolute encoder.

The position of the absolute encoder is read by an SSI link.

Principle Diagram

The figure below represents an SSI frame:



Data Information

The data content can be configured to adjust the information from the absolute Encoder:

Parameter	Range	Comment
Transmission speed	100 or 200 kHz	–
Length of frame	8...41 bits	Length of frame = implicit number of header bits (0 to 4) + number of data bits (8 to 32) + number of status bits (0 to 4) + number of parity bit (0 or 1).
Data	8...32 bits	Binary or gray code. The least significant bits (8…32) indicate resolution per turn and the most significant bits (0…24) indicate the number of turns.
Data bits per turn	8...16 bits	Binary or gray code.
Status bits	0...4 bits	–
Parity	0...1 bit	None, odd or even.
Resolution reduction	0...17 bits	This parameter allows to filter data. The least significant bits are ignored.
Error bit check	no check, active 0, active 1	• no check: the controller ignores any detected error generated by the encoder. • active 0: the controller ignores the frame containing the detected error. The active level is 0. • active 1: the controller ignores the frame containing the detected error. The active level is 1.

Section 19.2
Standard Encoder in SSI Mode on an Encoder Interface

Overview

This section describes the configuration in **SSI** mode of a **Motion Encoder** on the **Encoder** interface.

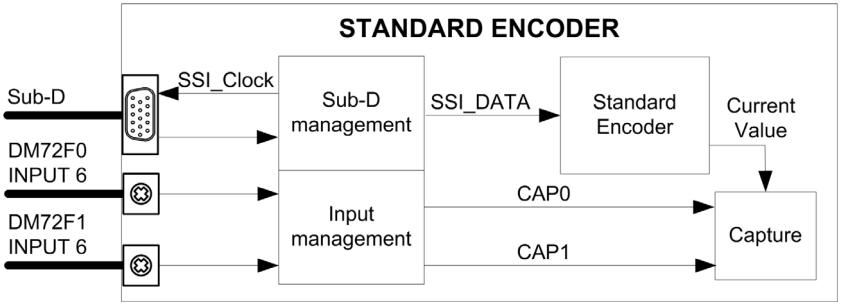
What Is in This Section?

This section contains the following topics:

Standard Encoder Specifications

Synopsis Diagram

The following diagram provides an overview of the **Standard Encoder** in **SSI** mode on the **Encoder** interface:



Optional Function

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

- Capture *(see page 189)*

Configuration of the Standard Encoder on an Encoder Interface

Configuration Window

Follow this procedure to configure the **Standard Encoder** type in **SSI** mode:

Step	Action
1	In the **Devices tree**, double-click **MyController → Expert → Encoder → Standard_Encoder_1.** **Result**: The configuration window is displayed.
2	Select the **Standard Encoder Configuration** tab.

Configuration Parameters

This table describes the properties of the **Standard Encoder Configuration** tab in **SSI** mode:

Parameter			Value	Default Value	Description
General	**Mode**		Incremental SSI	Incremental	Select the encoding mode.
Range	**Preset**		0	0	Set the counting initial value.
Capture	**CAP 0 input**	**Location**	Disabled I1.6	Disabled	Select the PLC input used for capturing the first counting value.
	CAP 1 input	**Location**	Disabled I3.6	Disabled	Select the PLC input used for capturing the second counting value.
Compare	**Thresholds**	**Number of thresholds**	0 1 2 3 4	0	Select the number of thresholds (0...4).
Scaling	**Increments**		2048	2048	Increment Per Turn.
	Units		360	360	Unit Per Turn.
Monitoring	**Power supply monitor**		Disabled Enabled	Disabled	Enable the power supply monitor.

Parameter		Value	Default Value	Description
Synchronous Serial Interface (SSI)	Transmission speed	100 200	200	Select the speed of the transmission (in KHz).
	Number of bits by frame	8	8	Set the number of bits by frame.
	Number of data bits	8	8	Set the number of bits to be reserved for the data.
	Number of data bits / turn	8	8	Set the number of data bits to be reserved to determine the number of turns.
	Number of status bits	0	0	Set the number of bits to be reserved for the status.
	Parity	None Even Odd	None	Select the parity.
	Resolution reduction	0	0	Set the resolution reduction.
	Binary coding	Binary Gray	Binary	Select the binary coding mode.

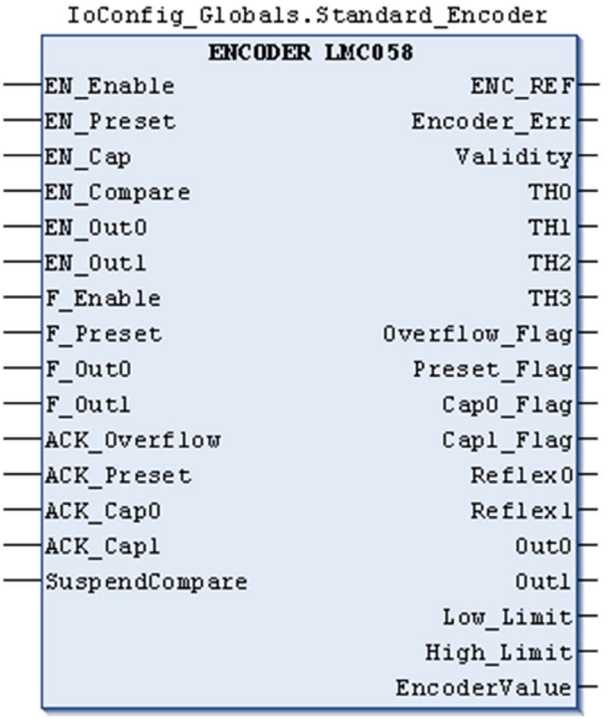
Programming the Standard Encoder

Overview

A **Standard Encoder** is always managed by an Encoder_LMC058 *(see page 232)* function block.

Adding a Standard Encoder Function Block

Step	Action
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → ENCODER → ENCODER_LMC058** in the list, drag-and-drop the item onto the **POU** window.
2	Type the `Encoder_LMC058` instance name or select the function block instance by clicking: `???` `...` Using the input assistant, the `Encoder_LMC058` 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	Not used.
EN_Preset	BOOL	Not used.
EN_Cap	BOOL	When at least one **CAP** input is configured, it authorizes the capture function via the inputs *(see page 189)*.
EN_Compare	BOOL	TRUE = enables the comparator operation using Thresholds 0, 1, 2, 3 *(see page 175)*: • basic comparison (TH0, TH1, TH2, TH3 output bits) • reflex (Reflex0, Reflex1 output bits) • events (to trigger external tasks on threshold crossing)
EN_Out0	BOOL	Not used.
EN_Out1	BOOL	Not used.
F_Enable	BOOL	Forces the Enable condition *(see page 198)*. In case of a detected SSI error, setting the F_Enable input to 0 acknowledges this detected error.
F_Preset	BOOL	Forces the Preset condition *(see page 194)*.
F_Out0	BOOL	Not used.
F_Out1	BOOL	Not used.
ACK_Overflow	BOOL	On rising edge, resets Overflow_Flag
ACK_Preset	BOOL	On rising edge, resets Preset_Flag *(see page 194)*.
ACK_Cap0	BOOL	On rising edge, resets Cap0_Flag *(see page 189)*.
ACK_Cap1	BOOL	On rising edge, resets Cap1_Flag *(see page 189)*.
SuspendCompare	BOOL	Not used.

The following table describes the output variables:

Outputs	Type	Comment
ENC_REF	EXPERT_REF *(see page 212)*	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 226)* 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 176)*.
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 (if configured) *(see page 176)*.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (if configured) *(see page 176)*.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (if configured) *(see page 176)*.
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 196)*.
Cap0_Flag	BOOL	Set to 1 when a new capture value is stored in the Capture register *(see page 189)*. 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 189)*. This flag must be reset before a new capture is allowed.
Reflex0	BOOL	State of Reflex0 *(see page 176)*.
Reflex1	BOOL	State of Reflex1 *(see page 176)*.
Out0	BOOL	Not relevant
Out1	BOOL	Not relevant
Low_Limit	BOOL	Not relevant
High_Limit	BOOL	Not relevant
EncoderValue	DINT	Current value of the encoder. The value is not valid if an error is detected.

Section 19.3
Motion Encoder in SSI Mode on the Encoder Interface

Overview

This section describes the configuration in **SSI** mode of a **Motion Encoder** on the **Encoder** interface.

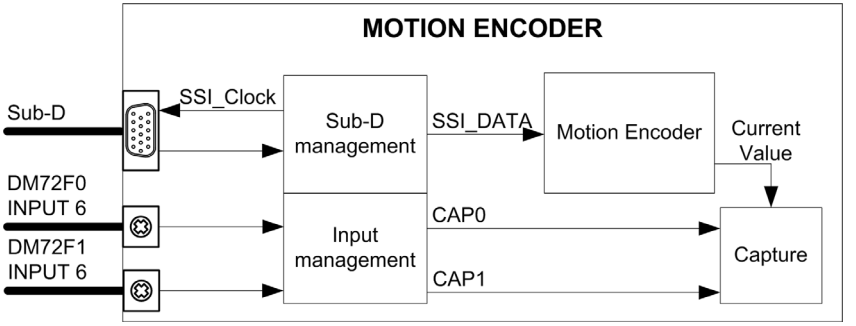
What Is in This Section?

This section contains the following topics:

Motion Encoder Specifications

Synopsis Diagram

The following diagram provides an overview of the **Motion Encoder** in **SSI** mode on the **Encoder** interface:



Optional Function

In addition to the **SSI** mode, the **Motion Encoder** can provide the following function:

- Capture *(see page 189)*

A **Motion Encoder** supports all SM3_Basic library SoftMotion functions (preset, touch prob, etc...).

In addition, a MC_GetImmediateValue_LMC058 *(see page 217)* and a MC_Reset_LMC058 *(see page 219)* functions are available in the **LMC058 Motion** library.

Configuration of the Motion Encoder on the Encoder Interface

Configuration Window

To configure the **Motion Encoder** type in **SSI** mode, double-click **MyController → Expert → Encoder → Motion_Encoder_1**.

Result: the **Motion Encoder Configuration** window is displayed.

Configuration Parameters

This table describes the properties of the **Motion Encoder Configuration** tab in **SSI** mode:

Parameter			Value	Default Value	Description
General	**Mode**		Incremental SSI	Incremental	Select the encoding mode.
Capture	**CAP 0 input**	**Location**	Disabled I1.6	Disabled	Select the PLC input used for capturing the first counting value.
	CAP 1 input	**Location**	Disabled I3.6	Disabled	Select the PLC input used for capturing the second counting value.
Monitoring	**Power supply monitor**		Disabled Enabled	Disabled	Enable the power supply monitor.
Synchronous Serial Interface (SSI)	**Transmission speed**		100 200	200	Select the speed of the transmission (in KHz).
	Number of bits by frame		8	8	Set the number of bits by frame.
	Number of data bits		8	8	Set the number of bits to be reserved for the data.
	Number of data bits / turn		8	8	Set the number of data bits to be reserved to determine the number of turns.
	Number of status bits		0	0	Set the number of bits to be reserved for the status.
	Parity		None Even Odd	None	Select the parity.
	Resolution reduction		0	0	Set the resolution reduction.
	Binary coding		Binary Gray	Binary	Select the binary coding mode.

Part IX
Optional Functions

Overview

This part provides information on optional functions for HSC.

What Is in This Part?

This part contains the following chapters:

Chapter 20
Comparison Function

Section 20.1
Comparison with a Main Type

Overview

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

What Is in This Section?

This section contains the following topics:

Comparison Principle with a Main type or an Encoder

Overview

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

- One-shot *(see page 41)*
- Modulo-loop *(see page 55)*
- Free-Large *(see page 75)*

The compare block with an Encoder manages Thresholds, Reflex outputs and Events.

Comparison is configured in the Configuration screen *(see page 182)* by activating at least one threshold.

Comparison can be used to trigger:

- a programming action on thresholds *(see page 179)*
- an event on a threshold associated with an external task *(see page 178)*
  NOTE: This option is only available for TM3XF• expansion modules, which support external events.
- reflex outputs *(see page 179)*.

Principle of a Comparison

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

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

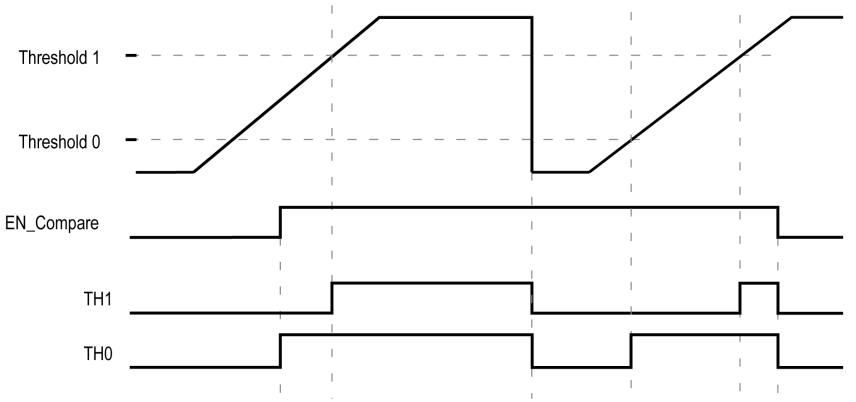
Threshold values are defined in the configuration window and can also be adjusted in the application program by using the EXPERTSetParam *(see page 230)* function block.

If Thresholdx (x= 0, 1, 2, 3) is configured and comparison is enabled (`EN_Compare` = 1), output pin THx of the `HSCMain_LMC058 (Encoder_LMC058)` function block is:

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

NOTE: When `EN_Compare` is set to 0 on `HSCMain_LMC058 (Encoder_LMC058)` function block, comparison functions are disabled, including external tasks triggered by a threshold event and Reflex outputs.

The following example for Modulo loop with two thresholds shows comparison in the `HSCMain_LMC058 (Encoder_LMC058)` function block:



Configuring Event Triggering in HSC Main Single or Dual Phase

Configuring an event on threshold crossing allows to trigger an external task *(see page 183)*. You can choose to trigger an event when a configured threshold is crossed as follows:

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

Configuring Event Triggering in Period Meter Mode

Configuring an event allows to trigger an external task *(see page 183)*. You can choose to trigger an event as follows:

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

Threshold Behavior

Using thresholds comparison status available in the task context (`TH0` to `TH2` output pins of the function block) is suitable for an application tolerant of the inherent lag of cycle times and asynchronism of communications, especially when using the modules over a field bus in distributed architectures.

Configuring Reflex Output

Follow this procedure to configure reflex outputs:

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

Reflex Output Behavior

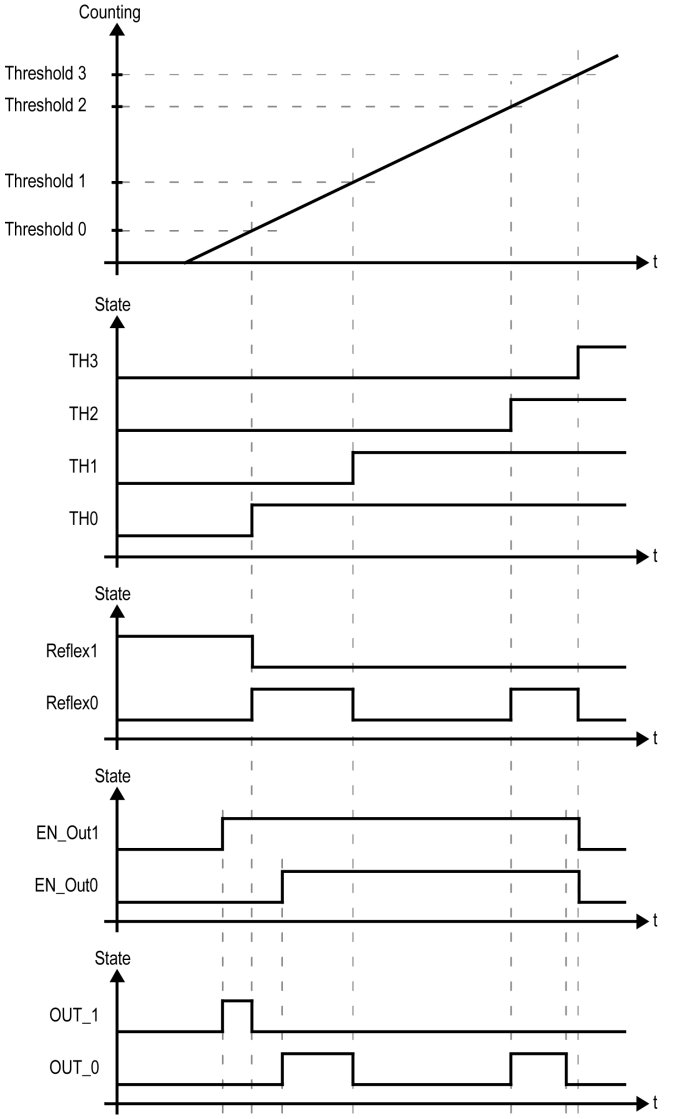
Configuring reflex outputs allows to trigger physical reflex outputs.

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

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

The performance is directly linked with the type of output used: fast or regular.

Example of the reflex outputs triggered by threshold:



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

Changing the Threshold Values

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

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

Step	Action
1	Set `SuspendCompare` to 1. The comparison is frozen at the counter value: ● The `TH0`, `TH1`, `Reflex0`, `Reflex1`, `Out0`, and `Out1` output bits of the function block maintain their last value. ● Physical outputs 0, 1 maintain their last value ● Events are masked **NOTE:** `EN_Compare`, `EN_Out0`, `EN_Out1`, `F_Out0`, and `F_Out1` remain operational while `SuspendCompare` is set.
2	Modify the threshold values as needed using the `EXPERTSetParam` *(see page 228)* function block. **NOTE:** Follow this rule to configure the threshold values: TH0 < TH1 < TH2 < TH3.
3	Set `SuspendCompare` to 0. The new threshold values are applied and the comparison is resumed.

Configuration of the Comparison on a Main Type or an Encoder

Configuration Procedure

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

Step	Action
1	In the **Devices tree**, double-click the **Expert → DM72F• → HSCMain** or **Standard_Encoder** node.
2	Select the **HSC Main Configuration** or **Standard Encoder Configuration** tab.
3	In the **Number of thresholds** parameter, select the number of thresholds to use.
4	Set the value of each threshold. **NOTE:** Follow this rule to configure the threshold values: TH0 < TH1 < TH2 < TH3
5	Optionally, configure the **Reflex Output** behavior *(see page 177)*. **Result:** External events in the selected group (HSCx_TH0, HSCx_TH1, HSCx_TH2, HSCx_TH3, HSCx_STOP) appear below **Threshold x External Event**.

External Event Configuration

Procedure

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

Step	Action
1	In the **Applications tree** tab, add a task.
2	Double-click the task node to associate it with to an external event.
3	In the **Type** dropdown menu, select **External**.
4	In the **External event** dropdown menu, select the event to associate to the task (see the 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_I0I4	Task is activated when the input I0 of the block **DM72F1** is set to 1.
BLOCK1_I1I5	Task is activated when the input I1 of the block **DM72F1** is set to 1.
BLOCK1_I2I6	Task is activated when the input I2 of the block **DM72F1** is set to 1.
BLOCK1_I3I7	Task is activated when the input I3 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.
BLOCK1_TH2	Task is activated when the threshold TH2 of the HSC or the encoder of the block **DM72F1** is set to 1.

Event Name	Description
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_HSCSTOP	Task is activated when the value of the HSC related to the block **DM72F0** reaches 0 in **One-shot** mode.
BLOCK1_HSCSTOP	Task is activated when the value of the HSC related to the block **DM72F1** reaches 0 in **One-shot** mode.
CAN0_SYNC	Task is activated when the CANopen manager sends a Sync message (enabled when **Enable Sync Producing** is checked in the CANopen manager configuration window).
CAN1_SYNC	Task is activated when the CANopen manager sends a Sync message (enabled when **Enable Sync Producing** is checked in the CANopen manager configuration window).

Chapter 21
Capture Function

Overview

This chapter provides information on capture function for HSC.

What Is in This Chapter?

This chapter contains the following sections:

Section 21.1
Capture with a Main Type

Overview

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

What Is in This Section?

This section contains the following topics:

Capture Principle with a Main Type

Overview

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

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

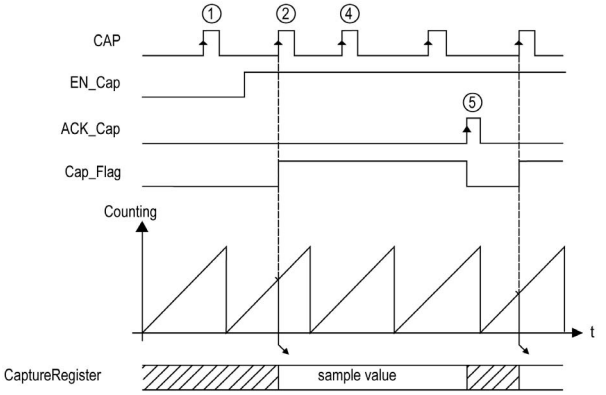
- One-shot *(see page 47)*
- Modulo-loop *(see page 67)*
- Free-large *(see page 83)*

To use this function:

- configure the optional Capture input **CAP**
- use the `EXPERTGetCapturedValue` *(see page 224)* function block to retrieve the captured value in your application.

Principle of a Capture

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



Stage	Action
1	When `EN_Cap` = 0, the function is not operational.
2	When `EN_Cap` = 1, the edge on **CAP** captures the counter value, puts it into the Capture register, and triggers the rising edge of `Cap_Flag`.
3	Get the stored value using `EXPERTGetCapturedValue` *(see page 224)*.
4	While `Cap_Flag` = 1, any new edge on the physical input CAP is ignored.
5	The rising edge of `HSCMain_LMC058` *(see page 235)* function block input `ACK_Cap` triggers the falling edge `Cap_Flag` output. A new capture is authorized.

Configuration of the Capture on a Main Type

Configuration Procedure

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

Step	Action
1	In the **Devices tree**, double-click the **Expert → DM72F• → HSCMain** or **Standard_Encoder** node.
2	Select the **HSC Main Configuration** or **Standard Encoder Configuration** tab.
3	Select a value for the **Capture → CAP input → Location** parameter.
4	Select a value for the **Capture → CAP input → Bounce filter** parameter.
5	Define the triggering mode of the **Capture → Mode** parameter.

Section 21.2
Capture with an Encoder

Overview

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

What Is in This Section?

This section contains the following topics:

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:

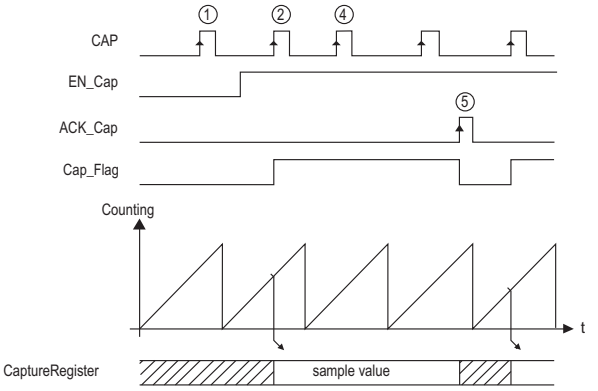
- Up to 2 position captures
- 1 distance capture

Using this function requires to:

- configure optional Capture inputs: **CAP**
- use `EXPERTGetCapturedValue` *(see page 224)* function block to retrieve the captured value in your application.

Principle of a Capture

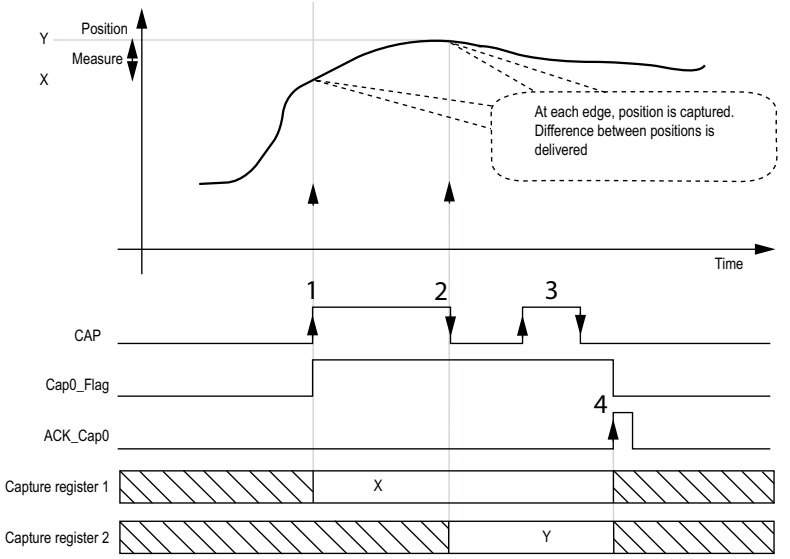
This graphic illustrates how the position capture works (only one register is shown):



Stage	Action
1	When `En_Cap` = 0, the function is not operational.
2	When `EN_Cap` = 1, the edge on **CAP** captures the current counter value and puts it into the Capture register, and triggers the rising edge of `Cap_Flag`.
3	Get the stored value using EXPERTGetCapturedValue *(see page 224)*.
4	While `Cap_Flag` = 1, any new edge on the physical input CAP is ignored.
5	The rising edge of `Encoder` *(see page 235)* function block input `ACK_Cap` triggers the falling edge `Cap_Flag` 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 224)*. The `EXPERTGetCapturedValue` *(see page 224)* function block can get: ● 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.
5	The rising edge of `Encoder` *(see page 235)* function block input `ACK_Cap` triggers the falling edge `Cap_Flag` 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 Procedure on an Expert Module

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

Step	Action
1	In the **Devices tree**, double-click the **Expert → DM72Fx → Standard_Encoder** or **Motion_Encoder**.
2	Select the configuration tab.
3	Select a value for the **Capture → CAP input → Location** parameter.
4	Select a value for the **Capture → CAP input → Bounce filter** parameter.
5	If necessary, select **CAP0 Mode** and **CAP1 Mode** parameters.
6	If necessary, enable the **Capture distance** parameter.

Configuration Procedure on an Encoder Interface

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

Step	Action
1	In the **Devices tree**, double-click the **Expert → Encoder → Standard_Encoder** or **Motion_Encoder**.
2	Select the configuration tab.
3	Select a value for the **Capture → CAP 0 input** or **CAP 1 input → Location** parameter.
4	Select a value for the **Capture → CAP 0 input** or **CAP 1 input → Bounce filter** parameter.
5	If necessary, select **CAP0 Mode** and **CAP1 Mode** parameters.
6	If necessary, enable the **Capture distance** parameter.

Chapter 22
Preset and Enable Functions

Overview

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

What Is in This Chapter?

This chapter contains the following topics:

Preset Function

Overview

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

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

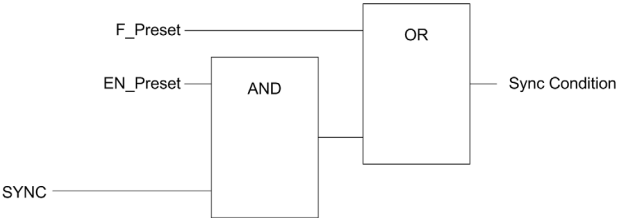
- **One shot** counter: preset and start the counter
- **Modulo-loop** counter: reset and start the counter
- **Event counting**: restart the internal time base at the beginning

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

Description

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

This diagram illustrates the Sync conditions of the HSC:



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

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

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

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

Configuration

This procedure describes how to configure a preset function:

Step	Action
1	In the **Devices tree**, double-click **MyController → DM72F• → HSCMain**.
2	Select the **HSC Main Configuration** tab.
3	Set the value of the **Counting function** parameter to **HSC Main Single Phase**, **HSC Main Dual Phase**, or **Period Meter**.
4	Select the value of the **Control inputs → SYNC input → Location** parameter.
5	Select the value of the **Control inputs → SYNC input → Bounce filter** parameter.
6	Select the value of the **Control inputs → Edge detection** parameter.

Transition Type

The transition type of the SYNC physical input is determined by the **Edge detection** parameter.

There are 3 available transitions, defined by configuration:

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

There are 3 available transitions, defined by configuration:

Value	Description
Sync Rising	Detection done on rising edge.
Sync Falling	Detection done on falling edge.
Sync Both	Detection done on rising edge and on falling edge.

Free-large or Period Meter Preset Conditions

Overview

In **Free-large** mode, the Preset condition is created by using two physical inputs:

- SYNC (respectively Z for the Encoder)
- REF

There are seven 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)

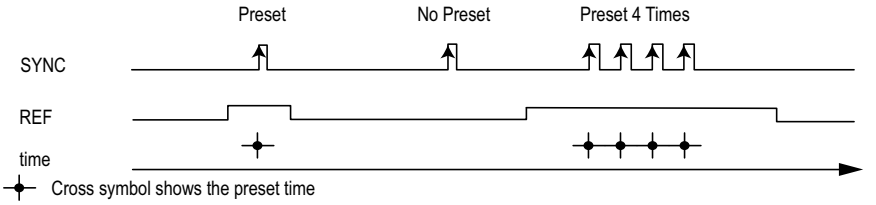
The counter synchronizes upon the encoder reference point.

At the Rising Edge of the Input REF

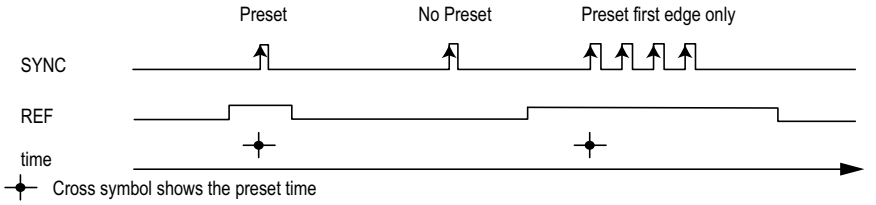
The counter synchronizes upon the mechanical position.

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

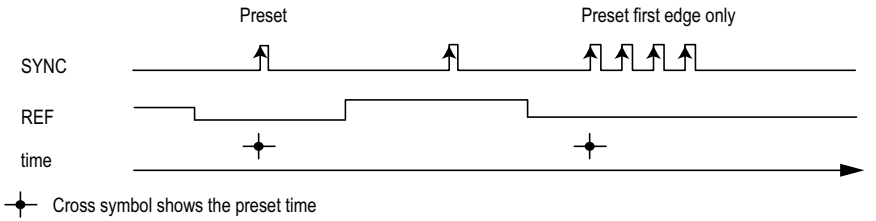
The counter synchronizes upon the encoder reference point when the **REF** signal is `TRUE`, as shown below:.



At the First SYNC Pulse after the REF Input Signal Rising:



At the first SYNC Pulse after the REF Input Signal Falling:



Enable: Authorize Counting Operation

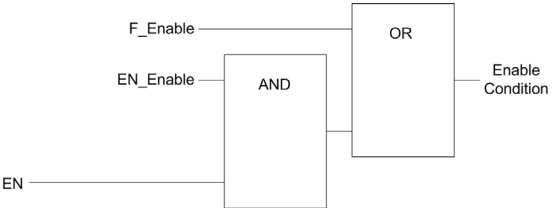
Overview

The enable function is used to authorize the counting operation.

Description

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

The following diagram illustrates the enable conditions:



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

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

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

Configuration

This procedure describes how to configure an enable function:

Step	Action
1	In the **Devices tree**, double-click **MyController → DM72F• → HSCMain**.
2	Select the **HSC Main Configuration** tab.
3	Select the value of the **Control inputs → EN input → Location** parameter.
4	Select the value of the **Control inputs → EN input → Bounce filter** parameter.

Appendices

Overview

The appendices provides an overview of data types, function blocks and general information about the function blocks used.

What Is in This Appendix?

The appendix contains the following chapters:

Appendix A
General Information

What Is in This Chapter?

This chapter contains the following topics:

Dedicated Features

Bounce Filter

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

Input	Bounce Filter Value (ms)	Maximum Counter Frequency Expert	Maximum Counter Frequency Regular
A B	0.000	200 kHz	1 kHz
	0.001	200 kHz	1 kHz
	0.002	200 kHz	1 kHz
	0.005	100 kHz	1 kHz
	0.01	50 kHz	1 kHz
	0.05	25 kHz	1 kHz
	0.1	5 kHz	1 kHz
	0.5	1 kHz	1 kHz
	1	500 Hz	500 Hz
	5	100 Hz	100 Hz
A is the counting input of the counter. B is the counting input of the dual phase counter.			

Dedicated Outputs

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

General Information on Administrative 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 into account.

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

Handling a Detected Error

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.

Appendix B
Data Types

Overview

This chapter describes the data types of the HSC Library.

What Is in This Chapter?

This chapter contains the following topics:

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.
EXPERT_UNKNOWN_PARAMETER	02 hex	The parameter reference is incorrect. See PARAMETER_TYPE section for valid parameters *(see page 211)*.
EXPERT_INVALID_PARAMETER	03 hex	The value of the parameter is incorrect. For example, Preset Value is <TH1 or <TH0.
EXPERT_COM_ERROR	04 hex	Communication error was detected with the EXPERT module.
EXPERT_CAPTURE_NOT_CONFIGURED	05 hex	Capture is not configured. It is impossible to get a captured value.

EXPERT_FREQMETER_TIMEBASE_TYPE: Type for Frequency Meter Time Base Variable

Enumerated Type Description

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

Name	Value
EXPERT_FREQMETER_10ms	10
EXPERT_FREQMETER_100ms	100
EXPERT_FREQMETER_1000ms	1000

EXPERT_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
EXPERT_IMMEDIATE_NO_ERROR	0	No error detected
EXPERT_IMMEDIATE_UNKNOWN	1	The reference of IMMEDIATE function is incorrect or not configured
IMMEDIATE_UNKNOWN_PARAMETER	2	A parameter reference is incorrect

MC_IMMEDIATE_ERR_TYPE: Type for Error Variable of the MC_GetImmediateValue_LMC058 Function Block

Enumerated Type Description

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

Enumerator	Value	Description
MC_IMMEDIATE_NO_ERROR	0	No error detected
MC_IMMEDIATE_UNKNOWN	1	The reference of IMMEDIATE function is incorrect or not configured
MC_IMMEDIATE_UNKNOWN_PARAMETER	2	A parameter reference is incorrect

EXPERT_PERIODMETER_RESOLUTION_TYPE: Type for Period Meter Time Base Variable

Enumerated Type Description

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

Name	Value
EXPERT_PERIODMETER_100ns	-1
EXPERT_PERIODMETER_1µs	1
EXPERT_PERIODMETER_100µs	100
EXPERT_PERIODMETER_1000µs	1000

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 *(see page 213)* 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 subparameters: **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
EXPERT_REFLEX1	0B hex	To get or set output 1 reflex mode of an EXPERT function

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 time base values allowed for use with an EXPERT function block:

Name	Value
EXPERT_100ms	100
EXPERT_1s	1000
EXPERT_10s	10000
EXPERT_60s	60000

Appendix C
Function Blocks

Overview

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

What Is in This Chapter?

This chapter contains the following sections:

Section C.1
LMC058 Motion Library

Overview

This section describes the functions in the **LMC058 Motion** library.

What Is in This Section?

This section contains the following topics:

MC_GetImmediateValue_LMC058: Read Counter Value of a MotionEncoder Function

Function Description

This function allows you to read the counter value of a Motion Encoder bypassing the controller cycle.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 241)*.

I/O Variables Description

The following table describes the input variables:

Input	Type	Comment
`Axis`	`AXIS_REF_SM3`	Reference the axis.

The following table describes the output variables:

Output	Type	Comment
`MCGetImmediateValue_LMC058`	`LREAL`	Contains the counter value.

The following table describes the output variables:

Input/Output	Type	Comment
Error	`BOOL`	`TRUE` = indicates that an error was detected. Function block execution is finished.
ErrID	`MC_IMMEDIATE_ERR_TYPE` *(see page 209)*	When `Error` is `TRUE`: type of the detected error.

Adding a MC_GetImmediateValue_LMC058

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**.
2	Select **Controller → LMC058 → LMC058 Motion → MC_GetImmediateValue_LMC058** in the list, drag-and-drop the item onto the POU window.
3	Link the `Axis` input to the `Axis_Ref` output of the Motion Encoder.

MC_Reset_LMC058: Clear the Detected Error on the SoftMotion Encoder

Function Description

This function allows you to clear the SoftMotion Encoder in case of an error detected. See online help Programming with EcoStruxure Machine Expert part, SoftMotion/Programming Interface/SoftMotion Libraries/SM3_Basic_library.

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 241)*.

I/O Variables Description

The following table describes the input variables:

Input	Type	Comment
`Axis`	`AXIS_REF_SM3`	Reference the axis.
`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:

Input	Type	Comment
`Done`	`BOOL`	Reference the axis.
`Busy`	`BOOL`	`TRUE` = indicates that the function block execution is in progress.
`Error`	`BOOL`	`TRUE` = indicates that an error was detected. Function block execution is finished.
`ErrorID`	`SMC_ERROR`	When `Error` is `TRUE`: type of the detected error.

Adding a MC_Reset_LMC058

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**.
2	Select **Controller → LMC058 → LMC058 Motion → MC_Reset_LMC058** in the list, drag-and-drop the item onto the POU window.
3	Link the `Axis` input to the `Axis_Ref` output of the Motion Encoder.

Section C.2
LMC058 EXPERT IO Library

Overview

This section describes the function blocks related to the HSC or the Standard Encoder of the **LMC058 EXPERT IO** library.

What Is in This Section?

This section contains the following topics:

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 *Function and Function Block Representation (see page 241)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF *(see page 212)*	Reference to the EXPERT function block.
Execute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.

This table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF *(see page 212)*	Reference to the EXPERT function block.
Done	BOOL	TRUE = indicates that ExpertDiag is valid. Function block execution is finished.
Error	BOOL	TRUE = indicates that an error was detected.
ErrID	IMMEDIATE_FUNC_ERR_TYPE *(see page 208)*	When Error is TRUE: type of the detected error.
ImmediateValue	DINT	Contains the counter value.

Adding the EXPERTGetImmediateValue Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → Administrative → EXPERTGetImmediateValue** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

EXPERTGetCapturedValue: Returns Content of Capture Registers

Function Block 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 *Function and Function Block Representation* *(see page 241)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
`EXPERT_REF_IN`	`EXPERT_REF` *(see page 212)*	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: • for HSCMain: always 0 • for Encoder: 0: Cap0, 1: Cap1, or 2: Distance

This table describes the output variables:

Outputs	Type	Comment
`EXPERT_REF_OUT`	`EXPERT_REF` *(see page 212)*	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.

Outputs	Type	Comment
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrID	EXPERT_ERR_TYPE *(see page 206)*	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 203)*.

Adding the EXPERTGetCapturedValue Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → Administrative → EXPERTGetCapturedValue** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

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

Function Block 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 *Function and Function Block Representation* *(see page 241)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
`EXPERT_REF_IN`	`EXPERT_REF` *(see page 212)*	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.

This table describes the output variables:

Outputs	Type	Comment
`EXPERT_REF_OUT`	`EXPERT_REF` *(see page 212)*	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.
`Error`	`BOOL`	`TRUE` = indicates that an error was detected. Function block execution is finished.

Outputs	Type	Comment
ErrID	EXPERT_ERR_TYPE *(see page 206)*	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 203)*.

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 198)* to 0 to acknowledge the error condition.		

Adding the EXPERTGetDiag Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → Administrative → EXPERTGetDiag** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

EXPERTGetParam: Returns Parameters of Principal EXPERT I/O Function

Function Block Description

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

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 241)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
`EXPERT_REF_IN`	`EXPERT_REF` *(see page 212)*	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 211)*	Parameter to read.

This table describes the output variables:

Outputs	Type	Comment
`EXPERT_REF_OUT`	`EXPERT_REF` *(see page 212)*	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.

Outputs	Type	Comment
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrID	EXPERT_ERR_TYPE *(see page 206)*	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 203)*.

Adding the EXPERTGetParam Function Block

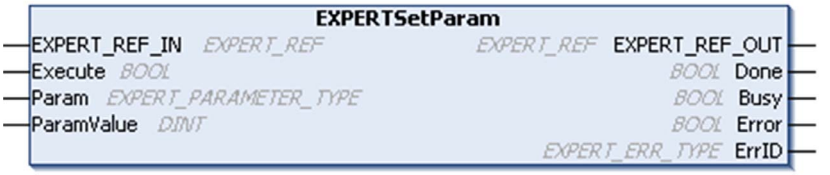
Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → Administrative → EXPERTGetParam** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

EXPERTSetParam: Adjust Parameters of a HSC

Function Block Description

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 241)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
`EXPERT_REF_IN`	`EXPERT_REF` *(see page 212)*	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 211)*	Parameter to read.
`ParamValue`	`DINT`	Parameter value to write.

This table describes the output variables:

Outputs	Type	Comment
`EXPERT_REF_OUT`	`EXPERT_REF` *(see page 212)*	Reference to the EXPERT function block.
`Done`	`BOOL`	`TRUE` = indicates that the parameter was successfully written. Function block execution is finished.

Outputs	Type	Comment
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 206)*	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 203)*.

Adding the EXPERTSetParam Function Block

Step	Description
1	Select the **Libraries** tab in the **Software Catalog** and click **Libraries**. Select **Controller → LMC058 → LMC058 Expert IO → Administrative → EXPERTSetParam** in the list, drag-and-drop the item onto the **POU** window.
2	Link the **EXPERT_REF_IN** input to the **HSC_REF** output of the HSC.

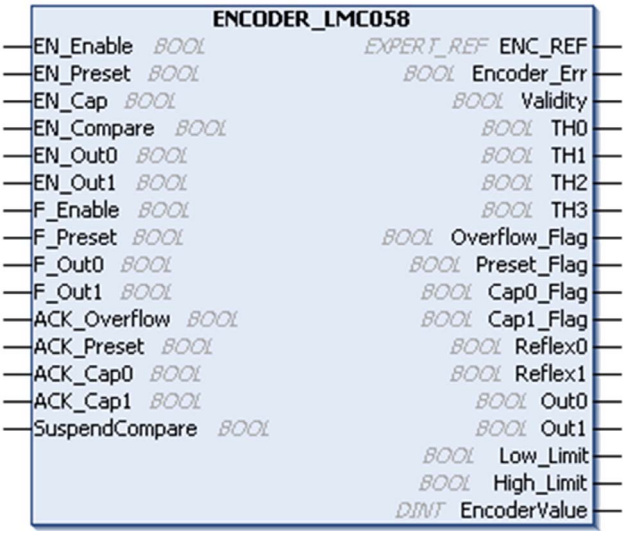
Encoder_LMC058: Encoder Function Block

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 *Function and Function Block Representation (see page 241)*.

I/O Variables Description

This 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): • 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 *(see page 198)*. In case of a detected SSI error setting the F_Enable input to 0 acknowledges the error.
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: • 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 SupendCompare is set.

This table describes the output variables:

Outputs	Type	Comment
ENC_REF	EXPERT_REF *(see page 212)*	Reference to the Encoder.
Encoder_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag *(see page 226)* 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 CurrentValue > Threshold 1 (if configured). Only active when EN_Compare is set.
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (if configured). Only active when EN_Compare is set.
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (if configured). Only active when EN_Compare 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 196)*.
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 Reflex0 (if configured). Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1 (if configured). Only active when EN_Compare 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: HSC Main Function Block

Function Block Description

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

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

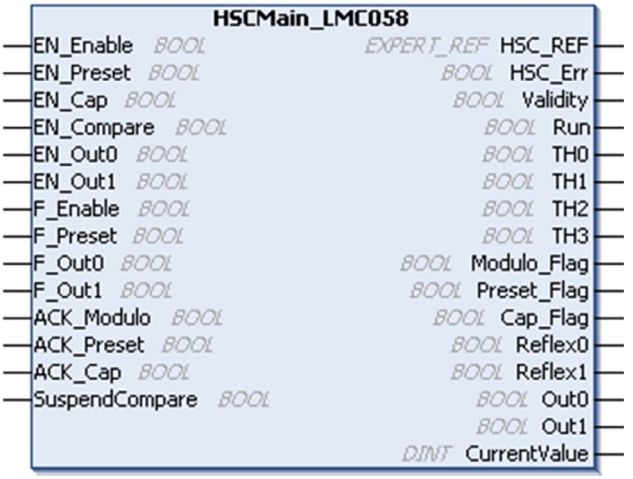
The HSC Main function block is mandatory when using **Main** counter.

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

⚠ WARNING
UNINTENDED OUTPUT VALUES ● Only use the Function Block instance in the MAST task. ● Do not use the same Function Block instance in a different task. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

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

Graphical Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation (see page 241)*.

I/O Variables Description

This table describes the input variables:

Input	Type	Description
`EN_Enable`	`BOOL`	`TRUE` = authorizes enabling of the counter using the `Enable` input.
`EN_Preset`	`BOOL`	`TRUE` = authorizes counter synchronization and start using the Sync input.
`EN_Cap`	`BOOL`	`TRUE` = enables the Capture input (if configured in **One shot**, **Modulo loop**, **Free large** modes).
`EN_Compare`	`BOOL`	`TRUE` = enables the comparator operation (using Thresholds 0, 1, 2, 3): • basic comparison (`TH0`, `TH1`, `TH2`, `TH3` output bits) • reflex (`Reflex0`, `Reflex1` 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).
`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` = activates counter and takes into account pulses on the counter input.

Input	Type	Description
F_Preset	BOOL	On rising edge, authorizes counting function synchronization and start in the following counting modes: **One-shot counter:** to preset and start the counter **Modulo loop counter:** to reset and start the counter **Free large counter:** to preset and start the counter **Event counter:** to restart the internal time base at the beginning **Frequency meter:** to restart the internal timer relative to the time base.
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 TRUE (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: • 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. • Compare events are masked. **NOTE:** EN_Compare, EN_Out0, EN_Out1, F_Out0, F_Out1 remain operational while SuspendCompare is set.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	Reference to the HSC.
Error	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag *(see page 226)* 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** Type, 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.

Outputs	Type	Comment
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 TRUE when the counter rolls over its limits in the following modes: ● **Modulo loop counter**: when the counter rolls over to the modulo or 0 ● **Free large** counter: when the counter roll overs its limits
Preset_Flag	BOOL	Set to TRUE by the synchronization of: ● **One-shot** counter: when the counter presets and starts ● **Modulo loop** counter: when the counter resets ● **Free large** counter: when the counter presets ● **Event** counter: when the internal timer relative to the time base restarts ● **Frequency meter**: when the internal timer relative to the time base restarts
Cap_Flag	BOOL	TRUE = indicates that a value has been latched in the capture register. This flag must be reset before a new capture can occur.
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	The value of the counter.

HSCSimple_LMC058: Control a Simple Type Counter for LMC058

Function Block Description

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

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

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

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

⚠ WARNING

UNINTENDED OUTPUT VALUES

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

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

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

Graphical Representation

IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* *(see page 241)*.

I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
Enable	BOOL	TRUE = activates counter and takes into account pulses on the counter input.
Sync	BOOL	On rising edge, loads the preset of the counter.
ACK_Modulo	BOOL	Modulo loop mode: On rising edge, resets the modulo flag Modulo_Flag.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF *(see page 212)*	Reference to the HSC.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag *(see page 226)* 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 activated. 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	Module loop mode: Set to TRUE when the counter rolls over the modulo value.
CurrentValue	DWORD	The value of the counter.

Appendix D
Function and Function Block Representation

Overview

Each function can be represented in the following languages:

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

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

What Is in This Chapter?

This chapter contains the following topics:

Differences Between a Function and a Function Block

Function

A function:

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

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

Function Block

A function block:

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

Examples: timers, counters

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

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

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

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

General Information

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

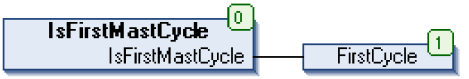
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

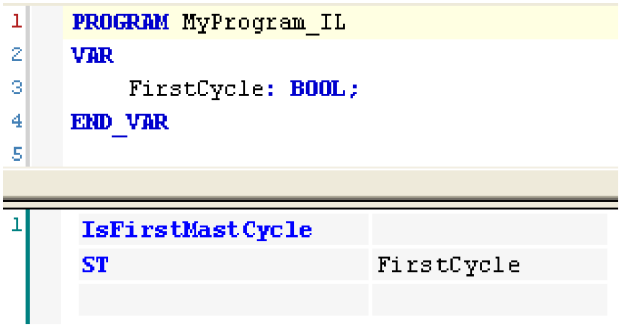
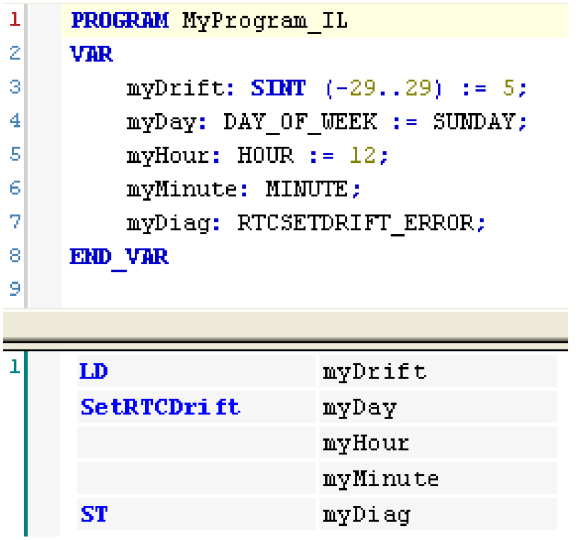
This procedure describes how to insert a function in IL language:

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

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

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

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

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

Using a Function Block in IL Language

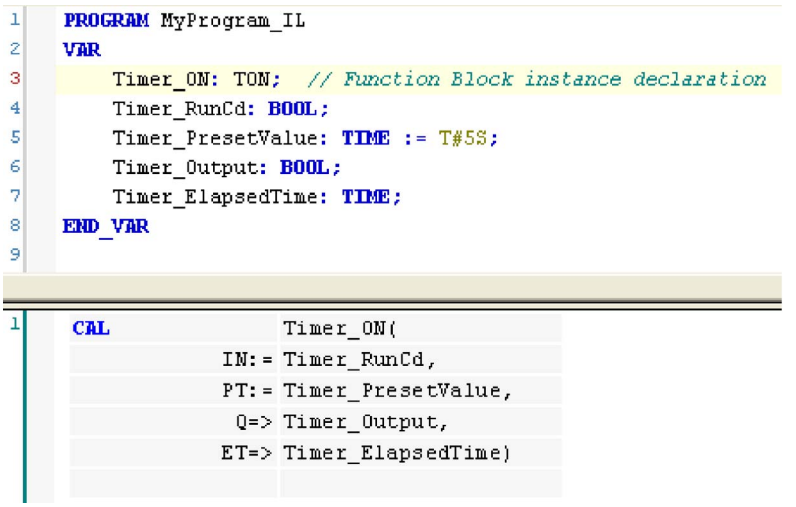
This procedure describes how to insert a function block in IL language:

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

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

Function Block	Graphical Representation
`TON`	

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

Function Block	Representation in POU IL Editor
TON	

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

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

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

General Information

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

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

Using a Function in ST Language

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

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

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

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

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

Using a Function Block in ST Language

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

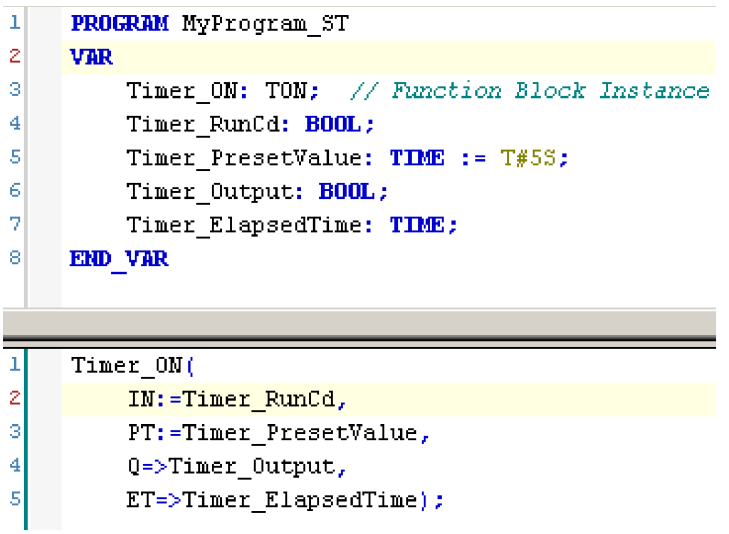
Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: • Input variables are the input parameters required by the function block • Output variables receive the value returned by the function block

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

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

Function Block	Graphical Representation
`TON`	

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

Function Block	Representation in POU ST Editor
TON	

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

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

Glossary

A

application

A program including configuration data, symbols, and documentation.

B

byte

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

C

CAN

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

CANmotion

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

CFC

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

control network

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

Two kinds of topologies are supported:

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

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

controller

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

CPDM

(*controller power distribution module*) The connection of the controller to the external 24 Vdc power supplies and the beginning of the power distribution for the local configuration.

E

encoder

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

F

FB

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

function

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

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

function block

A programming unit that has 1 or more inputs and returns 1 or more outputs. FBs are called through an instance (function block copy with dedicated name and variables) and each instance has a persistent state (outputs and internal variables) from 1 call to the other.

Examples: timers, counters

function block diagram

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

H

hex

(*hexadecimal*)

HSC

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

I

I/O

(*input/output*)

ID

(*identifier/identification*)

IEC 61131-3

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

IL

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

INT

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

L

LD

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

LMC058

Lexium motion controller A Modicon logic controller

N

network

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

node

An addressable device on a communication network.

P

POU

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

program

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

PTO

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

PWM

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

R

reflex output

Among the outputs of HSC are the reflex outputs associated to a threshold value that is compared to the counter value depending on the configuration of the HSC. The reflex outputs switch to either on or off depending on the configured relationship with the threshold.

run

A command that causes the controller to scan the application program, read the physical inputs, and write to the physical outputs according to solution of the logic of the program.

S

ST

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

STOP

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

T

task

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

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

A controller can have several tasks.

V

variable

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

Index

F

H

I

M

P

S