

# Modicon M241 Logic Controller

## High Speed Counting HSC Library Guide

04/2014



EIO0000001444.02

[www.schneider-electric.com](http://www.schneider-electric.com)

**Schneider**  
Electric

---

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

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

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

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

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

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

© 2014 Schneider Electric. All rights reserved.

---

# Table of Contents

---



	<b>Safety Information</b> .....	<b>7</b>
	<b>About the Book</b> .....	<b>9</b>
<b>Part I</b>	<b>Introduction</b> .....	<b>11</b>
<b>Chapter 1</b>	<b>High Speed Counter Overview</b> .....	<b>13</b>
	Expert I/O Overview .....	<b>14</b>
	Embedded Expert I/O Mapping .....	<b>16</b>
<b>Chapter 2</b>	<b>High Speed Counter Types</b> .....	<b>21</b>
	Choosing Your Counter .....	<b>22</b>
	<b>Simple Type Overview</b> .....	<b>26</b>
	<b>Main Type Overview</b> .....	<b>27</b>
	Frequency Meter Type Overview .....	<b>28</b>
	Period Meter Type Overview .....	<b>29</b>
<b>Part II</b>	<b>One-shot Mode</b> .....	<b>31</b>
<b>Chapter 3</b>	<b>One-shot Mode Principle</b> .....	<b>33</b>
	One-shot Mode Principle Description .....	<b>33</b>
<b>Chapter 4</b>	<b>One-shot with a Simple Type</b> .....	<b>35</b>
	Synopsis Diagram .....	<b>36</b>
	Configuration of the Simple Type in One-Shot Mode .....	<b>37</b>
	Programming the <b>Simple Type</b> .....	<b>38</b>
	Adjusting Parameters .....	<b>40</b>
<b>Chapter 5</b>	<b>One-shot With a Main Type</b> .....	<b>41</b>
	Synopsis Diagram .....	<b>42</b>
	Configuration of the Main Type Single Phase in One-Shot Mode .....	<b>43</b>
	Programming the <b>Main Type</b> .....	<b>44</b>
	Adjusting Parameters .....	<b>47</b>
<b>Part III</b>	<b>Modulo-loop Mode</b> .....	<b>49</b>
<b>Chapter 6</b>	<b>Modulo-loop Principle</b> .....	<b>51</b>
	Modulo-loop Mode Principle Description .....	<b>51</b>
<b>Chapter 7</b>	<b>Modulo-loop with a Simple Type</b> .....	<b>55</b>
	Synopsis Diagram .....	<b>56</b>
	Configuration of the Simple Type in Modulo-Loop Mode .....	<b>57</b>
	Programming the <b>Simple Type</b> .....	<b>58</b>
	Adjusting Parameters .....	<b>60</b>

---

<b>Chapter 8</b>	<b>Modulo-loop With a Main Type</b> .....	<b>61</b>
	Synopsis Diagram .....	<b>62</b>
	Configuration of the Main Type Single Phase in Modulo-Loop Mode .	<b>63</b>
	Configuration of the Main Type Dual Phase in Modulo-Loop Mode. . .	<b>64</b>
	Programming the <b>Main Type</b> .....	<b>65</b>
	Adjusting Parameters. ....	<b>68</b>
<b>Part IV</b>	<b>Free-large Mode</b> .....	<b>69</b>
<b>Chapter 9</b>	<b>Free-large Mode Principle</b> .....	<b>71</b>
	Free-large Mode Principle Description .....	<b>72</b>
	Limits Management .....	<b>75</b>
<b>Chapter 10</b>	<b>Free-large With a Main Type</b> .....	<b>77</b>
	Synopsis Diagram .....	<b>78</b>
	Configuration of the Main Type Dual Phase in Free-Large Mode . . . .	<b>79</b>
	Programming the <b>Main Type</b> .....	<b>80</b>
	Adjusting Parameters. ....	<b>83</b>
<b>Part V</b>	<b>Event Counting Mode</b> .....	<b>85</b>
<b>Chapter 11</b>	<b>Event Counting Principle</b> .....	<b>87</b>
	Event Counting Mode Principle Description .....	<b>87</b>
<b>Chapter 12</b>	<b>Event Counting With a Main Type</b> .....	<b>89</b>
	Synopsis Diagram .....	<b>90</b>
	Configuration of the Main Type Single Phase in Event Counting Mode	<b>91</b>
	Programming the <b>Main Type</b> .....	<b>92</b>
	Adjusting Parameters. ....	<b>95</b>
<b>Part VI</b>	<b>Frequency Meter Type</b> .....	<b>97</b>
<b>Chapter 13</b>	<b>Frequency Meter Principle</b> .....	<b>99</b>
	Description .....	<b>99</b>
<b>Chapter 14</b>	<b>Frequency Meter Type</b> .....	<b>101</b>
	Synopsis Diagram .....	<b>102</b>
	Configuration of the Frequency Meter Type .....	<b>103</b>
	Programming .....	<b>104</b>
<b>Part VII</b>	<b>Period Meter Type</b> .....	<b>107</b>
<b>Chapter 15</b>	<b>Period Meter Type Principle</b> .....	<b>109</b>
	Description .....	<b>109</b>

<b>Chapter 16</b>	<b>Period Meter Type</b> .....	<b>111</b>
	Synopsis Diagram .....	<b>112</b>
	Configuration of the Period Meter Type in Edge to Edge Mode .....	<b>113</b>
	Configuration of the Period Meter Type in Edge to Opposite Mode ..	<b>114</b>
	Programming .....	<b>115</b>
	Adjusting Parameters .....	<b>118</b>
<b>Part VIII</b>	<b>Optional Functions</b> .....	<b>119</b>
<b>Chapter 17</b>	<b>Comparison Function</b> .....	<b>121</b>
	Comparison Principle with a <b>Main</b> type .....	<b>122</b>
	Configuration of the Comparison on a <b>Main</b> Type .....	<b>126</b>
	External Event Configuration .....	<b>127</b>
<b>Chapter 18</b>	<b>Capture Function</b> .....	<b>129</b>
	Capture Principle with a <b>Main</b> Type .....	<b>130</b>
	Configuration of the Capture on a <b>Main</b> Type .....	<b>131</b>
<b>Chapter 19</b>	<b>Preset and Enable Functions</b> .....	<b>133</b>
	Preset Function .....	<b>134</b>
	Free-large or Period Meter Preset Conditions .....	<b>136</b>
	Enable Function .....	<b>138</b>
<b>Appendices</b>	.....	<b>139</b>
<b>Appendix A</b>	<b>General Information</b> .....	<b>141</b>
	Dedicated Features .....	<b>142</b>
	General Information on Administrative and Motion Function Block Management .....	<b>143</b>
<b>Appendix B</b>	<b>Data Types</b> .....	<b>145</b>
	EXPERT_ERR_TYPE: Type for Error Variable of EXPERT Function Block .....	<b>146</b>
	EXPERT_FREQMETER_TIMEBASE_TYPE: Type for Frequency Meter Time Base Variable .....	<b>147</b>
	EXPERT_HSCMAIN_TIMEBASE_TYPE: Type for HSC Main Time Base Variable .....	<b>148</b>
	EXPERT_IMMEDIATE_ERR_TYPE: Type for Error Variable of the GetImmediateValue Function Block .....	<b>149</b>
	EXPERT_PARAMETER_TYPE: Type for Parameters to Get or to Set on EXPERT .....	<b>150</b>
	EXPERT_PERIODMETER_RESOLUTION_TYPE: Type for Period Meter Time Base Variable .....	<b>151</b>
	EXPERT_REF: EXPERT Reference Value .....	<b>152</b>

---

<b>Appendix C</b>	<b>Function Blocks</b> .....	<b>153</b>
	EXPERTGetCapturedValue: Returns Content of Capture Registers . .	<b>154</b>
	EXPERTGetDiag: Provides Detail of Detected Error on a Principal	
	EXPERT I/O Function .....	<b>156</b>
	EXPERTGetImmediateValue: Read Counter Value of HSC .....	<b>158</b>
	EXPERTGetParam: Returns Parameters of Principal EXPERT I/O	
	Function .....	<b>160</b>
	EXPERTSetParam: Adjust Parameters of a HSC .....	<b>162</b>
	HSCMain_M241: HSC Main Function Block .....	<b>164</b>
	HSCSimple_M241: HSC Simple Function Block .....	<b>167</b>
<b>Appendix D</b>	<b>Function and Function Block Representation</b> .....	<b>169</b>
	Differences Between a Function and a Function Block .....	<b>170</b>
	How to Use a Function or a Function Block in IL Language .....	<b>171</b>
	How to Use a Function or a Function Block in ST Language. ....	<b>175</b>
<b>Glossary</b>	.....	<b>179</b>
<b>Index</b>	.....	<b>183</b>

---

# Safety Information

---



## Important Information

### NOTICE

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



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



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

## **DANGER**

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

## **WARNING**

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

## **CAUTION**

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

## **NOTICE**

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

---

## PLEASE NOTE

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

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



---

# About the Book

---



## At a Glance

### Document Scope

This documentation will acquaint you with the High Speed Counter (HSC) functions and variables offered within the M241 logic controller.

This documentation describes the functions and variables of the M241 HSC library.

In order to use this manual, you must:

- Have a thorough understanding of the M241, including its design, functionality, and implementation within control systems.
- Be proficient in the use of the following IEC 61131-3 PLC programming languages:
  - Function Block Diagram (FBD)
  - Ladder Diagram (LD)
  - Structured Text (ST)
  - Instruction List (IL)
  - Sequential Function Chart (SFC)

### Validity Note

This document has been updated with the release of SoMachine V4.1.

### Related Documents

Title of Documentation	Reference Number
SoMachine Programming Guide	EIO0000000067 (ENG), EIO0000000069 (FRE), EIO0000000068 (GER), EIO0000000071 (SPA), EIO0000000070 (ITA), EIO0000000072 (CHS)
Modicon M241 Logic Controller Programming Guide	EIO0000001432 (ENG), EIO0000001433 (FRE), EIO0000001434 (GER), EIO0000001435 (SPA), EIO0000001436 (ITA), EIO0000001437 (CHS)

You can download these technical publications and other technical information from our website at [www.schneider-electric.com](http://www.schneider-electric.com).

---

## Product Related Information

### **WARNING**

#### **LOSS OF CONTROL**

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.<sup>1</sup>
- 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.**

<sup>1</sup> 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.**

---

# Part I

## Introduction

---

### Overview

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

### What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
1	High Speed Counter Overview	13
2	High Speed Counter Types	21



---

# Chapter 1

## 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:

Topic	Page
Expert I/O Overview	14
Embedded Expert I/O Mapping	16

## Expert I/O Overview

### Introduction

The M241 logic controller provides:

I/O Type	24 I/O References	40 I/O References
	TM241•24•	TM241•40•
Fast inputs	8	8
Regular inputs	6	16
Fast outputs	4	4
Regular outputs	6	12

The M241 logic controller supports the following expert functions:

Functions		Description
Counters	HSC Simple	The HSC functions can execute fast counts of pulses from sensors, switches, and so on, that are connected to the fast inputs. For more information about the HSC functions, refer to the High Speed Counter types ( <a href="#">see page 21</a> ).
	HSC Main Single Phase	
	HSC Main Dual Phase	
	Frequency Meter	
	Period Meter	
Pulse Generators	PTO ( <i>see Modicon M241 Logic Controller, PTO PWM, Library Guide</i> )	The PTO function generates a pulse train output to control a linear single-axis stepper or servo drive in open loop mode.
	PWM ( <i>see Modicon M241 Logic Controller, PTO PWM, Library Guide</i> )	The PWM function generates a square wave signal on dedicated output channels with a variable duty cycle.
	Frequency Generator ( <i>see Modicon M241 Logic Controller, PTO PWM, Library Guide</i> )	The frequency generator function generates a square wave signal on dedicated output channels with a fixed duty cycle (50%).

#### NOTE:

- When an input is used as Run/Stop, it cannot be used by an expert function.
- When an output is used as Alarm, it cannot be used by an expert function.

For more details, refer to Embedded Functions Configuration (*see Modicon M241 Logic Controller, Programming Guide*).

## Configuring an Expert Function

To configure an expert function, proceed as follow:

Step	Description												
1	<p>Double-click the <b>Counters</b> or <b>Pulse_Generators</b> node in the <b>Devices Tree</b>.  <b>Result:</b> The <b>Counters</b> or <b>Pulse_Generators</b> function window appears:</p> <thead> <tr> <th>Parameter</th> <th>Type</th> <th>Value</th> <th>Default Value</th> <th>Unit</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>... Pulse generation function</td> <td>Enumeration of WORD</td> <td>None</td> <td>None</td> <td></td> <td>Select the pulse generation application</td> </tr> </tbody>	Parameter	Type	Value	Default Value	Unit	Description	... Pulse generation function	Enumeration of WORD	None	None		Select the pulse generation application
Parameter	Type	Value	Default Value	Unit	Description								
... Pulse generation function	Enumeration of WORD	None	None		Select the pulse generation application								

| 2 | Double-click **Value** and choose the function type to assign.  **Result:** The parameters of the expert function appear. |

## Expert Function I/O Within Regular I/O

Expert function I/O within regular I/O:

- Inputs can be read through standard memory variables even if configured as expert functions.
- An input cannot be configured as an expert function if it has already been configured as a Run/Stop input.
- An output cannot be configured in an expert function if it has already been configured as an alarm.
- Short-Circuit management still applies on all outputs. Status of outputs are available.
- All I/O that are not used by expert functions can be used as any other I/O.

When inputs are used in expert functions (Latch, HSC,...), integrator filter is replaced by anti-bounce filter. Filter value is configured in expert configuration screen.

## Embedded Expert I/O Mapping

### Input Mapping for Expert Functions on M241

M241 Expert Inputs		Fast Inputs							Regular Inputs						
		I0	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13
Event Latch 0	Input	M													
Event Latch 1	Input		M												
Event Latch 2	Input			M											
Event Latch 3	Input				M										
Event Latch 4	Input					M									
Event Latch 5	Input						M								
Event Latch 6	Input							M							
Event Latch 7	Input								M						
HSC Simple 0	Input	M													
HSC Simple 1	Input		M												
HSC Simple 2	Input			M											
HSC Simple 3	Input				M										
HSC Simple 4	Input					M									
HSC Simple 5	Input						M								
HSC Simple 6	Input							M							
HSC Simple 7	Input								M						
HSC Main 0	Input A	M													
	Input B/EN		C												
	SYNC			C											
	CAP				C							C			
	Reflex 0														
	Reflex 1														
HSC Main 1	Input A				M										
	Input B/EN					C									
	SYNC						C								
	CAP							C							C
	Reflex 0														
	Reflex 1														
<b>M</b> Mandatory															
<b>C</b> Depend on Configuration															



M241 Expert Inputs		Fast Inputs							Regular Inputs						
		I0	I1	I2	I3	I4	I5	I6	I7	I8	I9	I10	I11	I12	I13
Frequency Meter 0/Period Meter 0	Signal	M													
	EN		C												
Frequency Meter 1/Period Meter 1	Signal					M									
	EN						C								
PWM 0/Frequency Generator 0	Outputs														
	EN				C						C				
	SYNC									C					
PWM 1/Frequency Generator 1	Outputs														
	EN							C							C
	SYNC												C		
PTO 0	REF (Origin)								C						
	INDEX (Proximity)									C					
	PROBE				C						C				
	Output A/CW/Pulse														
	Output B/CCW/Dir														
PTO 1	REF (Origin)											C			
	INDEX (Proximity)												C		
	PROBE							C							C
	Output A/CW/Pulse														
	Output B/CCW/Dir														
<b>M</b> Mandatory															
<b>C</b> Depend on Configuration															

**Output Mapping for Expert Functions on M241**

M241 Expert Outputs		Fast Outputs				Regular Outputs					
		Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
HSC Main 0	Input A										
	Input B/EN										
	SYNC										
	CAP										
	Reflex 0	C				C					
	Reflex 1		C				C				
HSC Main 1	Input A										
	Input B/EN										
	SYNC										
	CAP										
	Reflex 0			C				C			
	Reflex 1				C				C		
Frequency Meter 0/Period Meter 0	Signal										
	EN										
Frequency Meter 1/Period Meter 1	Signal										
	EN										
PWM 0/Freq Gen 0	Outputs	M				C					
	EN										
	SYNC										
PWM 1/Freq Gen 1	Outputs			M				C			
	EN										
	SYNC										
PTO 0	REF (Origin)										
	INDEX (Proximity)										
	PROBE										
	Output A/CW/Pulse	M				C					
	Output B/CCW/Dir		M				C				
<b>M</b> Mandatory <b>C</b> Depend on Configuration											

M241 Expert Outputs		Fast Outputs				Regular Outputs					
		Q0	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9
PTO 1	REF (Origin)										
	INDEX (Proximity)										
	PROBE										
	Output A/CW/Pulse			M				C			
	Output B/CCW/Dir				M				C		
<b>M</b> Mandatory											
<b>C</b> Depend on Configuration											

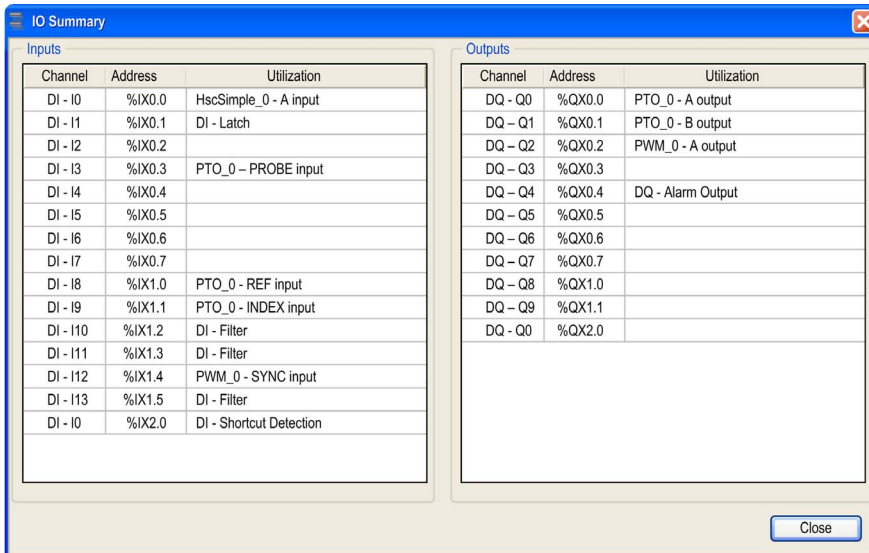
### I/O Summary

The **IO Summary** window displays the I/Os used by the expert functions.

To display the **IO Summary** window:

Step	Action
1	In the <b>Devices tree</b> tab, right-click the <b>MyController</b> node and choose <b>IO Summary</b> .

Example of IO Summary window:





---

# Chapter 2

## High Speed Counter Types

---

### Overview

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

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Choosing Your Counter	22
<b>Simple</b> Type Overview	26
<b>Main</b> Type Overview	27
Frequency Meter Type Overview	28
Period Meter Type Overview	29

## Choosing Your Counter

### Overview

Start the HSC configuration by choosing a counter according to the type of sensor you are using and the application need.

In the **Counters** editor, select a **Counting function** from the list that offers the following 5 types of counters (for more information, refer to the Counter Function (see *Modicon M241 Logic Controller, Programming Guide*):

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

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

For each counter function block, the **Instance name** is automatically given by SoMachine. The **Instance name** is editable and allows you to define the instance name. However, whether the **Instance name** is software-defined or user-defined, use the same instance name as an input to the function blocks dealing with the counter, as defined in the **Counters** editor.

### Type and Mode Matrix

This table presents the different types and modes available:

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

### HSC Simple

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

Feature	Function	
	One-shot Mode	Modulo-loop Mode
Counting mode	Count down	Count up
Enable with an HSC physical input	No	No
Synchronization / preset with an HSC physical input	No	No
Compare function	No	No
Capture function	No	No
Configuration tuning	–	–

### HSC Main Single Phase

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

Feature	Function		
	One-shot Mode	Modulo-loop Mode	Event Counting Mode
Counting mode	Count down	Count up	Pulse counting during given time base
Enable with an HSC physical input	Yes	Yes	No
Synchronization / preset with an HSC physical input	Yes	Yes	Yes
Compare function	Yes, 4 thresholds, 2 outputs, and 4 events	Yes, 4 thresholds, 2 outputs, and 4 events	No
Capture function	Yes, 1 capture register	Yes, 1 capture register	No
Configuration tuning	Stop event	–	Time base

### HSC Main Dual Phase

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

Feature	Function	
	Modulo-loop Mode	Free-Large Mode
Counting mode	Count up / down Pulse / direction Quadrature	Count up / down Pulse / direction Quadrature
Enable with an HSC physical input	No	No
Synchronization / preset with an HSC physical input	Yes	Yes
Compare function	Yes, 4 thresholds, 2 outputs, and 4 events	Yes, 4 thresholds, 2 outputs, and 4 events
Capture function	Yes, 1 capture register	Yes, 1 capture register
Configuration tuning	–	Limits management

### Frequency Meter

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

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



## Period Meter

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

Feature	Function	
	Edge to Edge Mode	Edge to Opposite Mode
Counting mode	Duration counting with the configurable resolution (1, 100, or 1000 $\mu$ s).	Duration counting with the configurable resolution (1, 100, or 1000 $\mu$ s).
Enable with an HSC physical input	Yes	Yes
Synchronization / preset with an HSC physical input	No	No
Compare function	No	No
Capture function	No	No
Configuration tuning	Resolution Time out	Resolution Time out

## Simple Type Overview

### Overview

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

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

With the **Simple** type, you cannot trigger an event or a reflex output.

### Simple Type Modes

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

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

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

### Performance

The maximum frequency admissible on an **Expert I/O** interface is 100 kHz if the bounce filter value is 0.005 ms (default value for configuration). If the bounce filter value is 0.002 ms, the maximum frequency is 200 kHz.

## Main Type Overview

### Overview

The **Main** type is a counter that uses up to 4 fast inputs and 2 reflex outputs. The M241 Logic Controller can have up to 2 **Main** type High Speed Counters.

### Main Type Modes

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

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

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

**Free-large** (*see page 77*): In this mode, the counter behaves like a high range up and down counter.

**Event Counting** (*see page 89*): In this mode, the counter accumulates a number of events that are received during a user-configured time base.

### Optional Features

Optional features can be configured depending on the selected mode:

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

### Performance

The maximum frequency admissible on an **Expert I/O** interface is 100 kHz if the bounce filter value is 0.005 ms (default value for configuration). If the bounce filter value is 0.002 ms, the maximum frequency is 200 kHz.

## Frequency Meter Type Overview

### Overview

The **Frequency Meter** type is a counter that uses up to 2 fast inputs. The M241 Logic Controller can have up to 2 **Frequency Meter** type High Speed Counters.

### Frequency Meter Type Mode

The **Frequency Meter** type supports the following counting mode:

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

### Performance

The maximum frequency admissible on an **Expert I/O** interface is 200 kHz.

## Period Meter Type Overview

### Overview

The **Period Meter** type is a counter that uses up to 2 fast inputs. The M241 Logic Controller can have up to 2 **Period Meter** type High Speed Counters.

### Period Meter Type Mode

The **Period Meter** type supports the following counting mode:

**Period meter** (*see page 111*): 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

### Performance

The maximum frequency admissible on an **Expert I/O** interface is 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	Chapter Name	Page
3	<b>One-shot</b> Mode Principle	33
4	<b>One-shot</b> with a <b>Simple</b> Type	35
5	<b>One-shot</b> With a <b>Main</b> Type	41





---

# Chapter 3

## One-shot Mode Principle

---

### One-shot Mode Principle Description

#### Overview

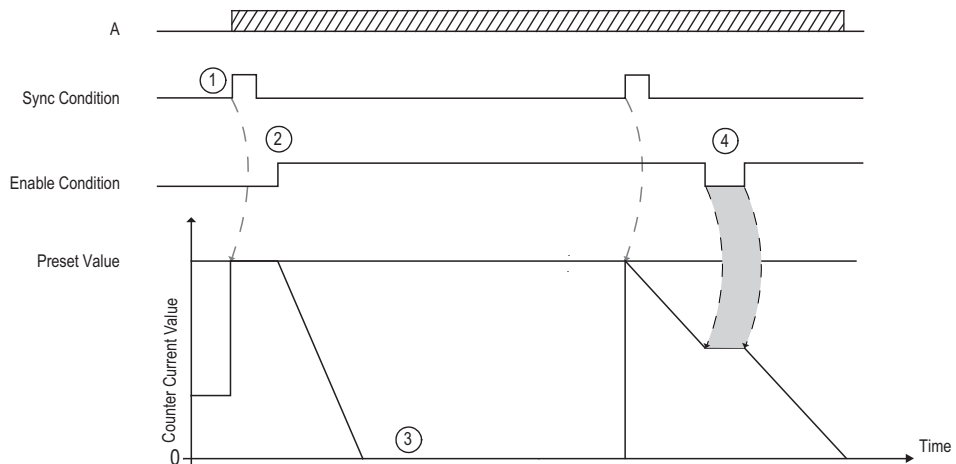
The counter is activated by a synchronization edge, and the preset value is loaded.

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

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

A new synchronization is needed to activate the counter again.

#### Principle Diagram



This table explains the stages from the preceding graphic:

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

**NOTE:** Enable and Sync conditions depends on configuration. These are described in the Enable ([see page 138](#)) and Preset ([see page 134](#)) function.

---

# Chapter 4

## One-shot with a Simple Type

---

### Overview

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

### What Is in This Chapter?

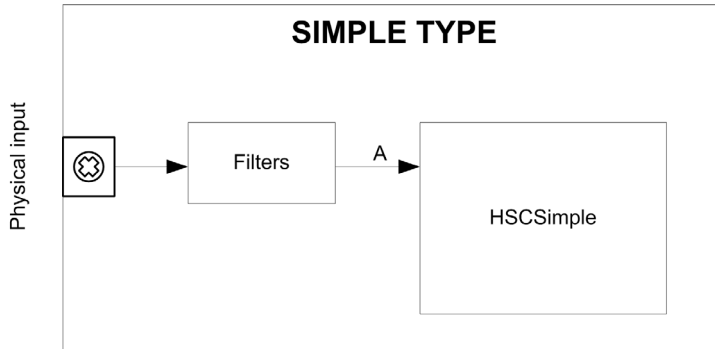
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	36
Configuration of the Simple Type in One-Shot Mode	37
Programming the <b>Simple</b> Type	38
Adjusting Parameters	40

## 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 down.

## Configuration of the Simple Type in One-Shot Mode

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>HSC Simple</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>General</b> → <b>Counting Mode</b> parameter to <b>One-shot</b> .
5	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <i>see page 142</i> ).
6	Enter the value of the <b>Range</b> → <b>Preset</b> parameter to set the counting initial value.


## Programming the Simple Type

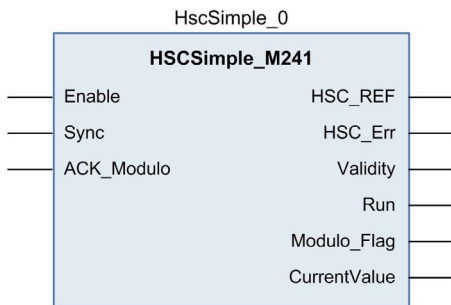
### Overview

A **Simple** type is always managed by an HSCSimple (see page 167) function block.

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

### Adding a HSCSimple Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>HSC</b> → <b>HSCSimple_M241</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Type the <b>Simple</b> 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: <b>Global Variables</b> → <b>MyController</b> → <b>PLC Logic</b> → <b>Config_Globals</b> .



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

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF (see page 152)	Reference to the HSC. To be used as input of <b>Administrative</b> function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (see page 156) 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	Set to 1 when the counter is running. Switches to 0 when CurrentValue reaches 0. A synchronization is needed to restart the counter.
Modulo_Flag	BOOL	Not relevant
CurrentValue	DWORD	Current count value of the counter.

## Adjusting Parameters

### Overview

The list of parameters described in the table can be read or modified by using the `EXPERTGetParam` (see page 160) or `EXPERTSetParam` (see page 162) function blocks.

**NOTE:** Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M241 Logic Controller, Programming Guide*).

### Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 150) that can be read or modified while the program is running:

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



---

# Chapter 5

## One-shot With a Main Type

---

### Overview

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

### What Is in This Chapter?

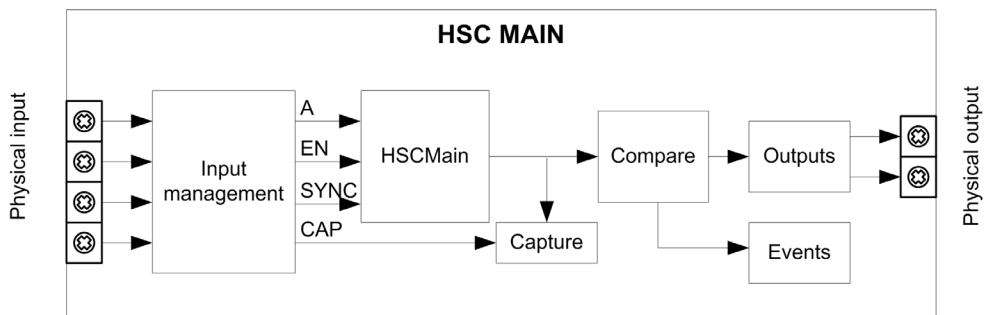
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	42
Configuration of the Main Type Single Phase in One-Shot Mode	43
Programming the <b>Main</b> Type	44
Adjusting Parameters	47

## 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.

CAP is the capture input of the counter.

SYNC is the synchronization input of the counter.

### Optional Function

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

- Comparison function (*see page 121*)
- Capture function (*see page 129*)
- Preset function (*see page 134*)
- Enable function (*see page 138*)

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

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>HSC Main Single Phase</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>General</b> → <b>Counting Mode</b> parameter to <b>One-shot</b> .
5	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <i>see page 142</i> ).
6	Enter the value of the <b>Range</b> → <b>Preset</b> parameter to set the counting initial value.
7	Optionally, you can enable these functions: <ul style="list-style-type: none"> <li>● Comparison function (<i>see page 121</i>)</li> <li>● Capture function (<i>see page 129</i>)</li> <li>● Preset function (<i>see page 134</i>)</li> <li>● Enable function (<i>see page 138</i>)</li> </ul>
8	Optionally, set the value of the <b>Events</b> → <b>Stop Event</b> parameter to <b>Yes</b> to enable the External Event Function ( <i>see page 127</i> ).


## Programming the Main Type

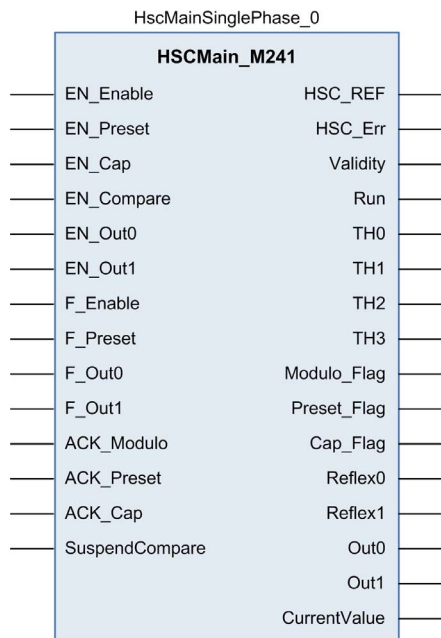
### Overview

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

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

### Adding the HSCMain Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>HSC</b> → <b>HSCMain_M241</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Type the <b>Main</b> 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: <b>Global Variables</b> → <b>&lt;MyController&gt;</b> → <b>PLC Logic</b> → <b>IoConfig_Globals</b> .



## 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 <b>EN</b> input is configured: if TRUE, authorizes the counter enable via the Enable input ( <i>see page 138</i> ).
EN_Preset	BOOL	When <b>SYNC</b> or <b>REF</b> input is configured: if TRUE, authorizes the counter Preset via the Sync input ( <i>see page 134</i> ).
EN_Cap	BOOL	When <b>CAP</b> input is configured: if TRUE, enables the Capture input.
EN_Compare	BOOL	TRUE = enables the comparator operation ( <i>see page 121</i> ) (using Thresholds 0, 1, 2, 3): <ul style="list-style-type: none"> <li>● basic comparison (TH0, TH1, TH2, TH3 output bits)</li> <li>● reflex (Reflex0, Reflex1 output bits)</li> <li>● events (to trigger external tasks on threshold crossing)</li> </ul>
EN_Out0	BOOL	TRUE = enables physical output Out_R0 to echo the Reflex0 value (if configured).
EN_Out1	BOOL	TRUE = enables physical output Out_R1 to echo the Reflex1 value (if configured).
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, presets and starts the counter.
F_Out0	BOOL	TRUE = forces Out_R0 to 1 (if Reflex0 is configured).
F_Out1	BOOL	TRUE = forces Out_R1 to 1 (if Reflex1 is configured).
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets Cap_Flag.
SuspendCompare	BOOL	TRUE = compare results are suspended: <ul style="list-style-type: none"> <li>● TH0, TH1, TH2, TH3, Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value.</li> <li>● Hardware Outputs 0, 1 maintain their last value.</li> <li>● Events are masked.</li> </ul> <p><b>NOTE:</b> EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.</p>

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF (see page 152)	Reference to the HSC. To be used as input of <b>Administrative</b> function blocks.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	TRUE = counter is running. The Run bit switches to 0 when CurrentValue reaches 0.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 (see page 121).
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 (see page 121).
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (see page 121).
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (see page 121).
Preset_Flag	BOOL	Set to 1 by the preset of the counter (see page 134).
Cap_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 can occur.
Reflex0	BOOL	State of Reflex0. Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1. Only active when EN_Compare is set.
Out0	BOOL	State of physical output Out_R0 (if Reflex0 configured).
Out1	BOOL	State of physical output Out_R1 (if Reflex1 configured).
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 160) or `EXPERTSetParam` (see page 162) function blocks.

**NOTE:** Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M241 Logic Controller, Programming Guide*).

### Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 150) which can be read or modified while the program is running:

Parameter	Description
<code>EXPERT_PRESET</code>	to get or set the Preset value of an HSC
<code>EXPERT_THRESHOLD0</code>	to get or set the Threshold 0 value of an HSC
<code>EXPERT_THRESHOLD1</code>	to get or set the Threshold 1 value of an HSC
<code>EXPERT_THRESHOLD2</code>	to get or set the Threshold 2 value of an HSC
<code>EXPERT_THRESHOLD3</code>	to get or set the Threshold 3 value of an HSC
<code>EXPERT_REFLEX0</code>	to get or set output 0 reflex mode of an EXPERT function
<code>EXPERT_REFLEX1</code>	to get or set output 0 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	Chapter Name	Page
6	<b>Modulo-loop</b> Principle	51
7	<b>Modulo-loop</b> with a <b>Simple</b> Type	55
8	<b>Modulo-loop</b> With a <b>Main</b> Type	61



---

# Chapter 6

## Modulo-loop Principle

---

### Modulo-loop Mode Principle Description

#### Overview

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

#### Principle

On a rising edge of the Sync condition (*see page 134*), the counter is activated and the current value is reset to 0.

When counting is enabled (*see page 138*):

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

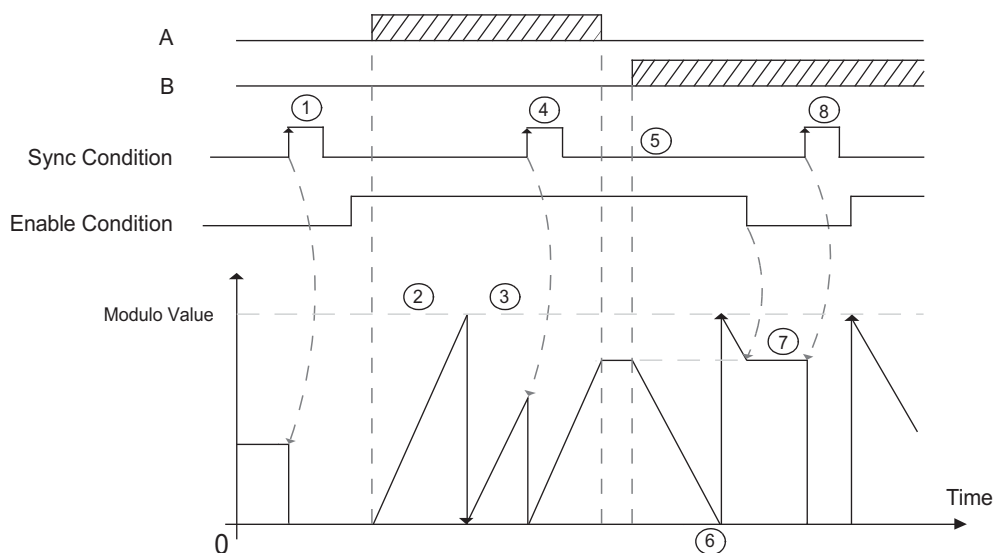
**Decrementing direction:** the counter decrements until it reaches 0. At the next pulse, the counter is set to the modulo value, a modulo flag is set to 1, and the counting continues.

#### 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: <ul style="list-style-type: none"><li>• X1: 1 count by Encoder cycle</li><li>• X2: 2 counts by Encoder cycle</li><li>• X4: 4 counts by Encoder cycle</li></ul>
Normal Quadrature X2	
Normal Quadrature X4	
Reverse Quadrature X1	
Reverse Quadrature X2	
Reverse Quadrature X4	

### Up Down Principle Diagram

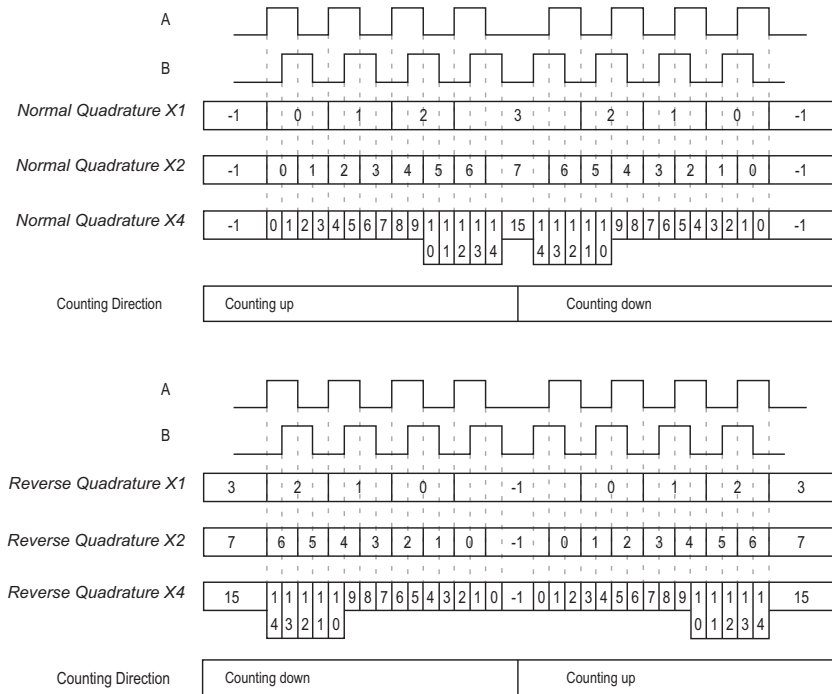


Stage	Action
1	On the rising edge of Sync condition, the current value is reset to 0 and the counter is activated.
2	When Enable condition = 1, each pulses on A increments the counter value.
3	When the counter reaches the (modulo-1) value, the counter loops to 0 at the next pulse and the counting continues.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

**NOTE:** Enable and Sync conditions depends on configuration. These are described in the Enable (see page 138) and Preset (see page 134) function.

### Quadrature Principle Diagram

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





---

# Chapter 7

## Modulo-loop with a Simple Type

---

### Overview

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

### What Is in This Chapter?

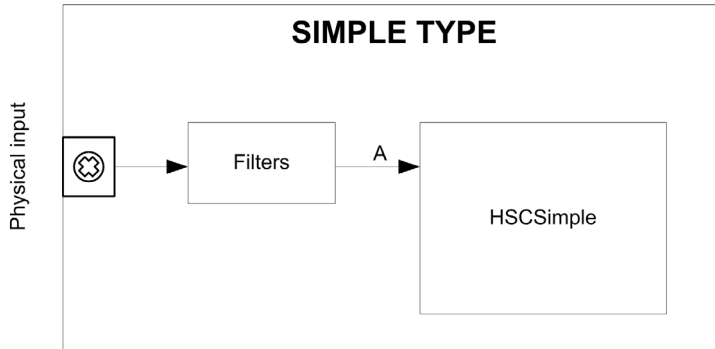
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	56
Configuration of the Simple Type in Modulo-Loop Mode	57
Programming the <b>Simple</b> Type	58
Adjusting Parameters	60

## Synopsis Diagram

### Synopsis Diagram

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



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



## Configuration of the Simple Type in Modulo-Loop Mode

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>HSC Simple</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>General</b> → <b>Counting Mode</b> parameter to <b>Modulo-loop</b> .
5	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <i>see page 142</i> ).
6	Enter the value of the <b>Range</b> → <b>Modulo</b> parameter to set the counting modulo value.


## Programming the Simple Type

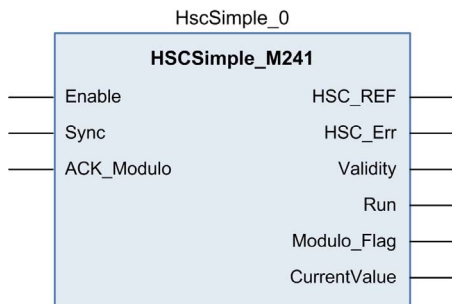
### Overview

A **Simple** type is always managed by an HSCSimple (see page 167) function block.

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

### Adding a HSCSimple Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>HSC</b> → <b>HSCSimple_M241</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Type the <b>Simple</b> 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: <b>Global Variables</b> → <b>&lt;MyController&gt;</b> → <b>PLC Logic</b> → <b>IoConfig_Globals</b> .



## 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 = authorizes changes to the current counter value.
Sync	BOOL	On rising edge, resets 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 ( <i>see page 152</i> )	Reference to the HSC. To be used as input of the <b>Administrative</b> function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag ( <i>see page 156</i> ) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that the output values on the function block are valid.
Run	BOOL	Not relevant
Modulo_Flag	BOOL	Set to 1 when the counter roll overs the modulo.
CurrentValue	DWORD	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 160) or `EXPERTSetParam` (see page 162) function blocks.

**NOTE:** Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M241 Logic Controller, Programming Guide*).

### Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 150) that can be read or modified while the program is running:

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

---

# Chapter 8

## Modulo-loop With a Main Type

---

### Overview

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

### What Is in This Chapter?

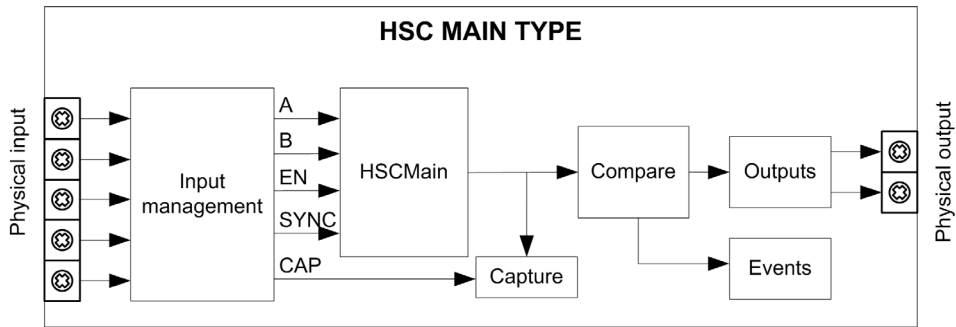
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	62
Configuration of the Main Type Single Phase in Modulo-Loop Mode	63
Configuration of the Main Type Dual Phase in Modulo-Loop Mode	64
Programming the <b>Main</b> Type	65
Adjusting Parameters	68

## 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.

CAP is the capture input of the counter.

SYNC is the synchronization input of the counter.

### Optional Function

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

- Comparison function (*see page 121*)
- Capture function (*see page 129*)
- Preset function (*see page 134*)
- Enable function (*see page 138*)

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

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>HSC Main Single Phase</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>General</b> → <b>Counting Mode</b> parameter to <b>Modulo-loop</b> .
5	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <i>see page 142</i> ).
6	Enter the value of the <b>Range</b> → <b>Modulo</b> parameter to set the counting modulo value.
7	Optionally, you can enable these functions: <ul style="list-style-type: none"> <li>● Comparison function (<i>see page 121</i>)</li> <li>● Capture function (<i>see page 129</i>)</li> <li>● Preset function (<i>see page 134</i>)</li> <li>● Enable function (<i>see page 138</i>)</li> </ul>

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

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>HSC Main Dual Phase</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>General</b> → <b>Counting Mode</b> parameter to <b>Modulo-loop</b> .
5	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <i>see page 142</i> ).
6	Set the value of the <b>Counting inputs</b> → <b>B input</b> → <b>Bounce filter</b> parameter.
7	Enter the value of the <b>Range</b> → <b>Modulo</b> parameter to set the counting modulo value.
8	Optionally, you can enable these functions: <ul style="list-style-type: none"> <li>● Comparison function (<i>see page 121</i>)</li> <li>● Capture function (<i>see page 129</i>)</li> <li>● Preset function (<i>see page 134</i>)</li> </ul>




## Programming the Main Type

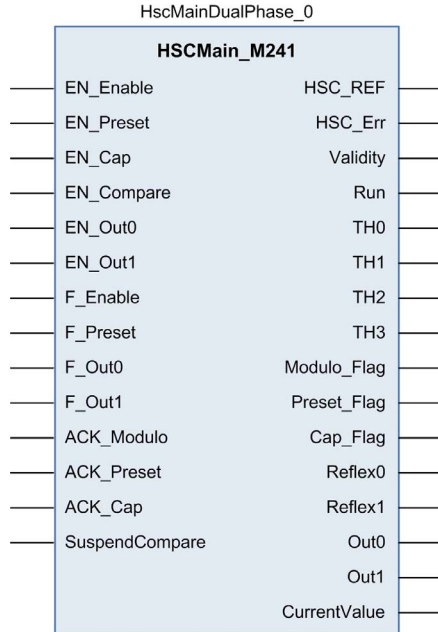
### Overview

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

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

### Adding the HSCMain Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>HSC</b> → <b>HSCMain_M241</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Type the <b>Main</b> 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: <b>Global Variables</b> → <b>&lt;MyController&gt;</b> → <b>PLC Logic</b> → <b>IoConfig_Globals</b> .



## 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 <b>EN</b> input is configured: if TRUE, authorizes the counter enable via the Enable input ( <i>see page 138</i> ).
EN_Preset	BOOL	When <b>SYNC</b> or <b>REF</b> input is configured: if TRUE, authorizes the counter Preset via the Sync input ( <i>see page 134</i> ).
EN_Cap	BOOL	When <b>CAP</b> input is configured: if TRUE, enables the Capture input.
EN_Compare	BOOL	TRUE = enables the comparator operation ( <i>see page 121</i> ) using Threshold 0, 1, 2, 3: <ul style="list-style-type: none"> <li>● basic comparison (TH0, TH1, TH2, TH3 output bits)</li> <li>● reflex (Reflex0, Reflex1 output bits)</li> <li>● events (to trigger external tasks on threshold crossing)</li> </ul>
EN_Out0	BOOL	TRUE = enables physical output Out_R0 to echo the Reflex0 value (if configured).
EN_Out1	BOOL	TRUE = enables physical output Out_R1 to echo the Reflex1 value (if configured).
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, resets, and starts the counter.
F_Out0	BOOL	TRUE = forces Out_R0 to 1 (if Reflex0 is configured).
F_Out1	BOOL	TRUE = forces Out_R1 to 1 (if Reflex1 is configured).
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets Cap_Flag.
SuspendCompare	BOOL	TRUE = compare results are suspended: <ul style="list-style-type: none"> <li>● TH0, TH1, TH2, TH3, Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value.</li> <li>● Hardware Outputs 0, 1 maintain their last value.</li> <li>● Events are masked.</li> </ul> <p><b>NOTE:</b> EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.</p>

This table describes the output variables:

Output	Type	Comment
HSC_REF	EXPERT_REF ( <i>see page 152</i> )	Reference to the HSC. To be used as input of <b>Administrative</b> function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag ( <i>see page 156</i> ) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	TRUE = counter is running. The Run bit switches to 0 when CurrentValue reaches 0. A synchronization is needed to restart the counter.
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 ( <i>see page 121</i> ).
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 ( <i>see page 121</i> ).
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 ( <i>see page 121</i> ).
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 ( <i>see page 121</i> ).
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 ( <i>see page 134</i> ).
Cap_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 can occur.
Reflex0	BOOL	State of Reflex0. Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1. Only active when EN_Compare is set.
Out0	BOOL	State of physical output Out_R0 (if Reflex0 is configured).
Out1	BOOL	State of physical output Out_R1 (if Reflex1 is configured).
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 160) or `EXPERTSetParam` (see page 160) function blocks.

**NOTE:** Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M241 Logic Controller, Programming Guide*).

### Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 150) that can be read or modified while the program is running:

Parameter	Description
<code>EXPERT_MODULO</code>	to get or set the Modulo value of an HSC
<code>EXPERT_THRESHOLD0</code>	to get or set the Threshold 0 value of an HSC
<code>EXPERT_THRESHOLD1</code>	to get or set the Threshold 1 value of an HSC
<code>EXPERT_THRESHOLD2</code>	to get or set the Threshold 2 value of an HSC
<code>EXPERT_THRESHOLD3</code>	to get or set the Threshold 3 value of an HSC
<code>EXPERT_REFLEX0</code>	to get or set output 0 reflex mode of an EXPERT function
<code>EXPERT_REFLEX1</code>	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	Chapter Name	Page
9	Free-large Mode Principle	71
10	Free-large With a Main Type	77



---

# Chapter 9

## Free-large Mode Principle

---

### Overview

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

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Free-large Mode Principle Description	72
Limits Management	75

## 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 138](#)), 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 136](#)) which loads the preset value.

The current counter is stored in the capture register by using the Capture ([see page 129](#)) function.

If the counter reaches the counting limits, the counter will react according to the Limits Management ([see page 75](#)) configuration.

### Input Modes

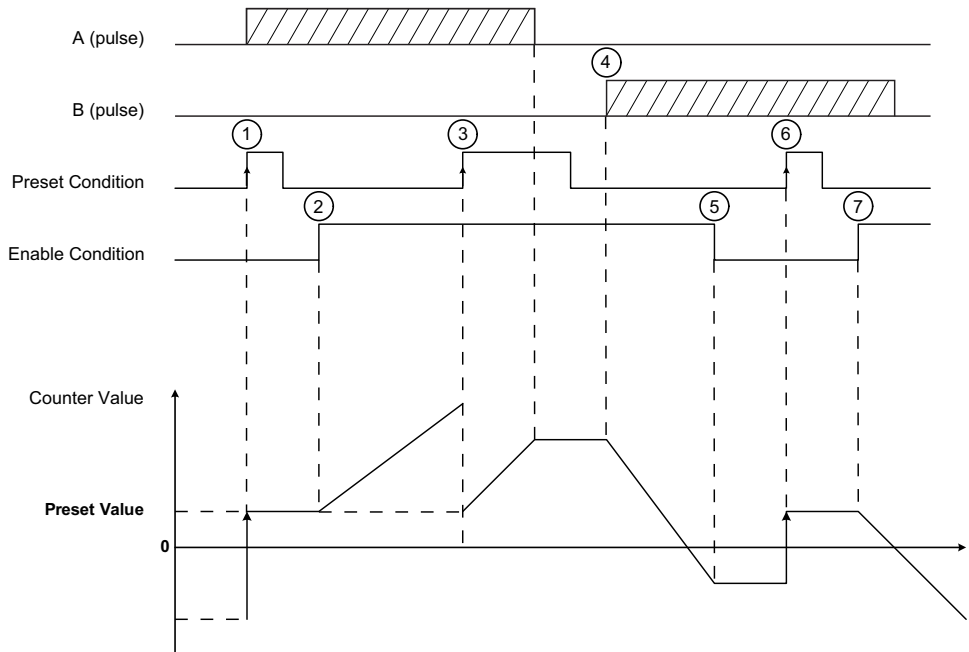
This table shows the 8 types of input modes available:

Input Mode	Comment
A = Up, B = Down	default mode The counter increments on A and decrements on B.
A = Impulse, B = Direction	If there is a rising edge on A and B is true, then the counter decrements. If there is a rising edge on A and B is false, then the counter increments.
Normal Quadrature X1	A physical encoder always provides 2 signals 90° shift that first allows the counter to count pulses and detect direction: <ul style="list-style-type: none"> <li>● X1: 1 count by Encoder cycle</li> <li>● X2: 2 counts by Encoder cycle</li> <li>● X4: 4 counts by Encoder cycle</li> </ul>
Normal Quadrature X2	
Normal Quadrature X4	
Reverse Quadrature X1	
Reverse Quadrature X2	
Reverse Quadrature X4	



## Up Down Principle Diagram

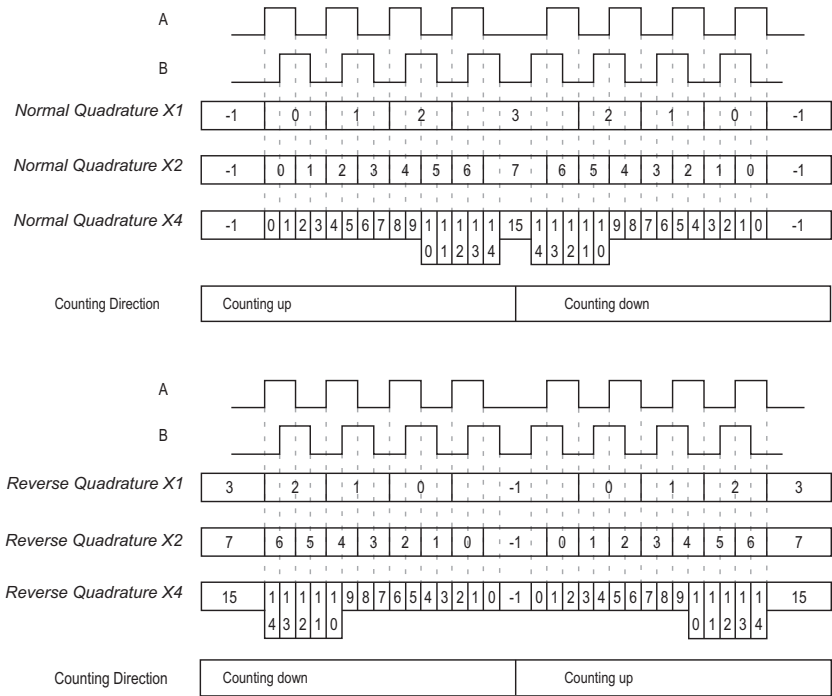
The figures shows the **A = Up, B = Down** mode:



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

### Quadrature Principle Diagram

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



## Limits Management

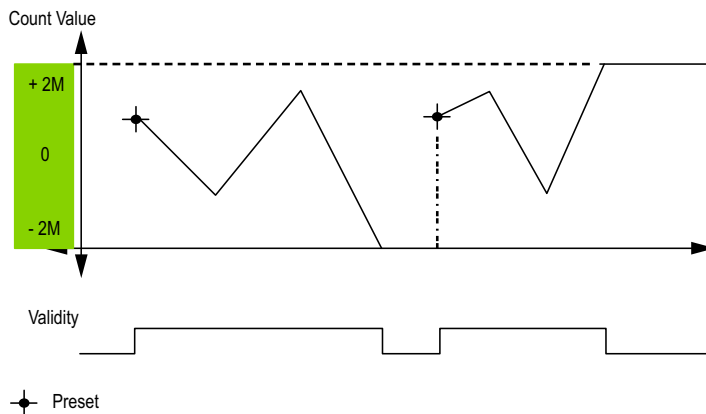
### Overview

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

- Lock on limits
- Rollover

### Lock on Limits

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



2M value is given as:

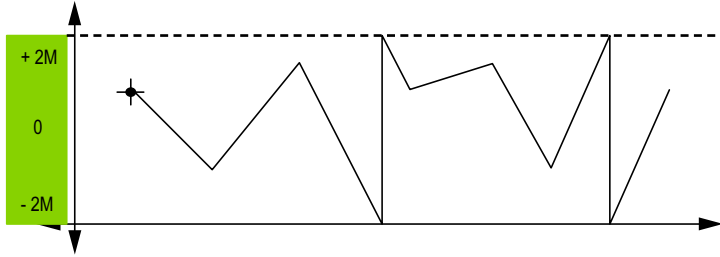
- $+2M = 2^{(\text{exp } 31)} - 1$
- $-2M = -2^{(\text{exp } 31)}$

### Rollover

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

Modulo\_Flag output is set to 1.

Count Value



---

# Chapter 10

## Free-large With a Main Type

---

### Overview

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

### What Is in This Chapter?

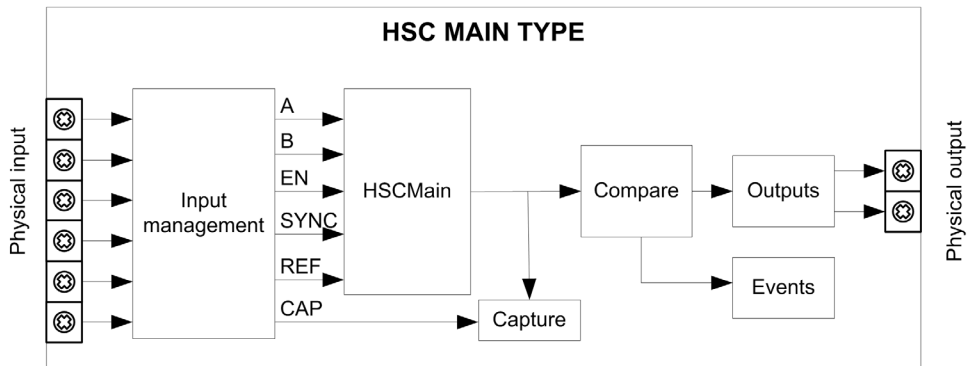
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	78
Configuration of the Main Type Dual Phase in Free-Large Mode	79
Programming the <b>Main</b> Type	80
Adjusting Parameters	83

## 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.

CAP is the capture input of the counter.

SYNC is the synchronization input of the counter.

### Optional Function

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

- Comparison function (*see page 121*)
- Capture function (*see page 129*)
- Preset function (*see page 134*)
- Enable function (*see page 138*)

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

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>HSC Main Dual Phase</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>General</b> → <b>Counting Mode</b> parameter to <b>Free-large</b> .
5	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <i>see page 142</i> ).
6	Set the value of the <b>Counting inputs</b> → <b>B input</b> → <b>Bounce filter</b> parameter.
7	Enter the value of the <b>Range</b> → <b>Preset</b> parameter to set the counting initial value.
8	Enter the value of the <b>Range</b> → <b>Limits</b> for limits management.
9	Optionally, you can enable these functions: <ul style="list-style-type: none"> <li>● Comparison function (<i>see page 121</i>)</li> <li>● Capture function (<i>see page 129</i>)</li> <li>● Preset function (<i>see page 134</i>)</li> </ul>


## Programming the Main Type

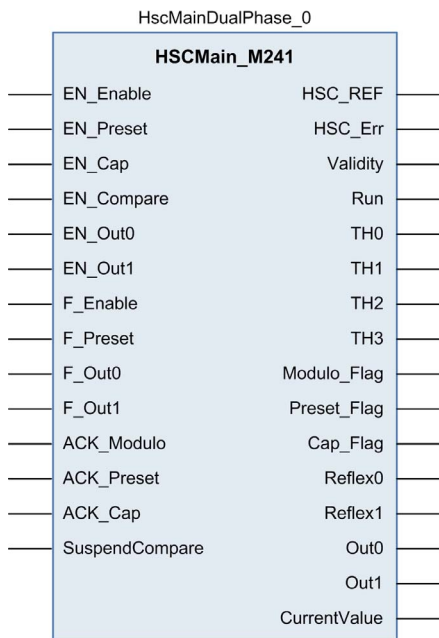
### Overview

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

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

### Adding the HSCMain Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>HSC</b> → <b>HSCMain_M241</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Type the <b>Main</b> 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: <b>Global Variables</b> → <b>&lt;MyController&gt;</b> → <b>PLC Logic</b> → <b>IoConfig_Globals</b> .





## 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 <b>EN</b> input is configured: if TRUE, authorizes the counter enable via the Enable input ( <a href="#">see page 138</a> ).
EN_Preset	BOOL	When <b>SYNC</b> or <b>REF</b> input is configured: if TRUE, authorizes the counter Preset via the Sync input ( <a href="#">see page 134</a> ).
EN_Cap	BOOL	When <b>CAP</b> input is configured: if TRUE, enables the Capture input.
EN_Compare	BOOL	TRUE = enables the comparator operation ( <a href="#">see page 121</a> ) (using Thresholds 0, 1, 2, 3): <ul style="list-style-type: none"> <li>● basic comparison (TH0, TH1, TH2, TH3 output bits)</li> <li>● reflex (Reflex0, Reflex1 output bits)</li> <li>● events (to trigger external tasks on threshold crossing)</li> </ul>
EN_Out0	BOOL	TRUE = enables physical output Out_R0 to echo the Reflex0 value (if configured).
EN_Out1	BOOL	TRUE = enables physical output Out_R1 to echo the Reflex1 value (if configured).
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, presets and starts the counter.
F_Out0	BOOL	TRUE = forces Out_R0 to 1 (if Reflex0 is configured).
F_Out1	BOOL	TRUE = forces Out_R1 to 1 (if Reflex1 is configured).
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag.
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets Cap_Flag.
SuspendCompare	BOOL	TRUE = compare results are suspended: <ul style="list-style-type: none"> <li>● TH0, TH1, TH2, TH3, Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value.</li> <li>● Hardware Outputs 0, 1 maintain their last value.</li> <li>● Events are masked.</li> </ul> <p><b>NOTE:</b> EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.</p>

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF (see page 152)	Reference to the HSC. To be used as input of <b>Administrative</b> function blocks.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the EXPERTGetDiag (see page 156) function block to get more information about this detected error.
Validity	BOOL	TRUE = indicates that output values on the function block are valid.
Run	BOOL	Not relevant
TH0	BOOL	Set to 1 when CurrentValue > Threshold 0 (see page 121).
TH1	BOOL	Set to 1 when CurrentValue > Threshold 1 (see page 121).
TH2	BOOL	Set to 1 when CurrentValue > Threshold 2 (see page 121).
TH3	BOOL	Set to 1 when CurrentValue > Threshold 3 (see page 121).
Modulo_Flag	BOOL	Set to 1 when the counter rollovers its limits.
Preset_Flag	BOOL	Set to 1 by the preset of the counter (see page 134)
Cap_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 can occur.
Reflex0	BOOL	State of Reflex0. Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1. Only active when EN_Compare is set.
Out0	BOOL	State of physical outputs Out_R0 (if Reflex0 configured).
Out1	BOOL	State of physical outputs Out_R1 (if Reflex1 configured).

## Adjusting Parameters

### Overview

The list of parameters described in the table can be read or modified by using the `EXPERTGetParam` (see page 160) or `EXPERTSetParam` (see page 162) function blocks.

**NOTE:** Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M241 Logic Controller, Programming Guide*).

### Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 150) which can be read or modified while the program is running:

Parameter	Description
<code>EXPERT_PRESET</code>	to get or set the Preset value of the HSC
<code>EXPERT_THRESHOLD0</code>	to get or set the Threshold 0 value of an HSC
<code>EXPERT_THRESHOLD1</code>	to get or set the Threshold 1 value of an HSC
<code>EXPERT_THRESHOLD2</code>	to get or set the Threshold 2 value of an HSC
<code>EXPERT_THRESHOLD3</code>	to get or set the Threshold 3 value of an HSC
<code>EXPERT_REFLEX0</code>	to get or set output 0 reflex mode of an expert function
<code>EXPERT_REFLEX1</code>	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	Chapter Name	Page
11	<b>Event Counting</b> Principle	87
12	<b>Event Counting</b> With a <b>Main</b> Type	89



---

# Chapter 11

## Event Counting Principle

---

### Event Counting Mode Principle Description

#### Overview

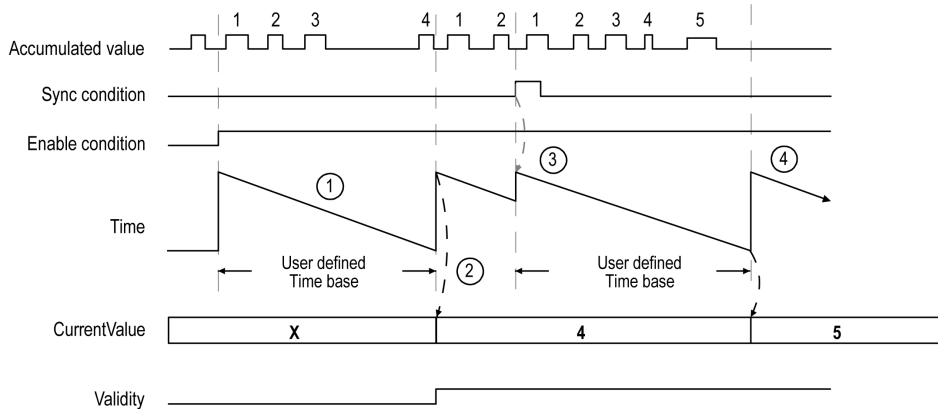
The **Event Counting** mode allows you to count a sequence of events during a given period of time.

#### Principle

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

The synchronization can be used over the time period. This restarts the counting event for a new predefined time period. The counting restarts at the edge Sync condition ([see page 134](#)).

Principle Diagram



Stage	Action
1	When Enable condition = 1, the counter accumulates the number of events (pulses) on the physical input during a predefined period of time. If Validity = 0, the current value is not relevant.
2	Once the first period of time has elapsed, the counter value is set to the number of events counted over the period and Validity is set to 1. The counting restarts for a new period of time.
3	On the rising edge of the Sync condition: <ul style="list-style-type: none"> <li>the accumulated value is reset to 0</li> <li>the current value is not updated</li> <li>the counting restarts for a new period of time</li> </ul>
4	Once the period of time has elapsed, the counter value is set to the number of events counted over the period. The counting restarts for a new period of time.

**NOTE:**

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

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



---

# Chapter 12

## Event Counting With a Main Type

---

### Overview

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

### What Is in This Chapter?

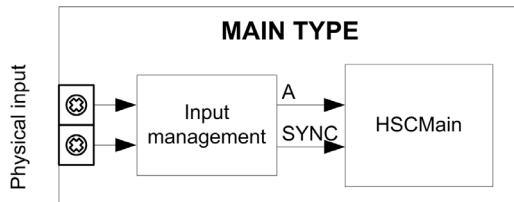
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	90
Configuration of the Main Type Single Phase in Event Counting Mode	91
Programming the <b>Main</b> Type	92
Adjusting Parameters	95

## Synopsis Diagram

### Synopsis Diagram

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



A is the counting input of the counter.

SYNC is the synchronization input of the counter.

### Optional Function

In addition to the **Event Counting** mode, the **Main** type provides the Preset function (*see page 134*).

## Configuration of the Main Type Single Phase in Event Counting Mode

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>HSC Main Single Phase</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>General</b> → <b>Counting Mode</b> parameter to <b>Event Counting</b> .
5	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <i>see page 142</i> ).
6	Set the value of the <b>Range</b> → <b>Time base</b> parameter to determine the period during which the number of events is counted. Select the measurement of the update cycle time: <ul style="list-style-type: none"> <li>● 0.1 s</li> <li>● 1 s (default value)</li> <li>● 10 s</li> <li>● 60 s</li> </ul>
7	Optionally, set the value of the <b>Control inputs</b> → <b>SYNC input</b> → <b>Location</b> parameter to enable the Preset Function ( <i>see page 134</i> ).


## Programming the Main Type

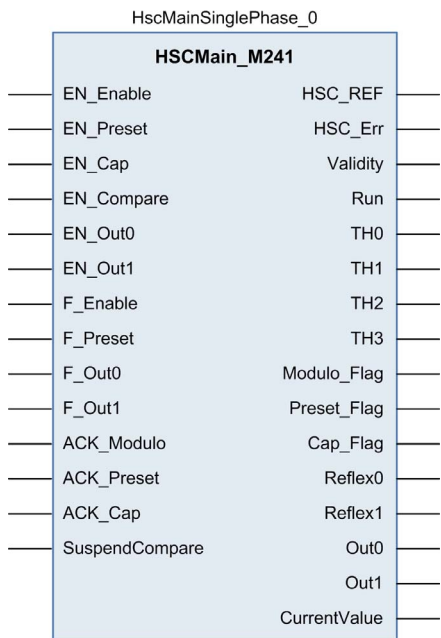
### Overview

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

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

### Adding the HSCMain Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>HSC</b> → <b>HSCMain_M241</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Type the <b>Main</b> 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: <b>Global Variables</b> → <b>&lt;MyController&gt;</b> → <b>PLC Logic</b> → <b>IoConfig_Globals</b> .



## I/O Variables Usage

These tables describes 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	Not used
EN_Preset	BOOL	When <b>SYNC</b> or <b>REF</b> input is configured: if <b>TRUE</b> , authorizes the counter Preset via the Sync input ( <i>see page 134</i> ).
EN_Cap	BOOL	Not used
EN_Compare	BOOL	Not used
EN_Out0	BOOL	Not used
EN_Out1	BOOL	Not used
F_Enable	BOOL	<b>TRUE</b> = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, restarts the internal timer relative to the time base.
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 <b>NOTE:</b> EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF ( <i>see page 152</i> )	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the <b>Administrative</b> function blocks.
HSC_Err	BOOL	<b>TRUE</b> = indicates that an error was detected. EXPERTGetDiag ( <i>see page 156</i> ) function block may be used to get more information about this detected error.
Validity	BOOL	<b>TRUE</b> = indicates that output values on the function block are valid.
Run	BOOL	Not relevant
TH0	BOOL	Not relevant
TH1	BOOL	Not relevant
TH2	BOOL	Not relevant
TH3	BOOL	Not relevant

Outputs	Type	Comment
Modulo_Flag	BOOL	Not relevant
Preset_Flag	BOOL	Set to 1 by the preset of the counter ( <i>see page 134</i> )
Cap_Flag	BOOL	Not relevant
Reflex0	BOOL	Not relevant
Reflex1	BOOL	Not relevant
Out0	BOOL	Not relevant
Out1	BOOL	Not relevant
CurrentValue	DINT	Current 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 160) or `EXPERTSetParam` (see page 162) function blocks.

**NOTE:** Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M241 Logic Controller, Programming Guide*).

### Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 150) which can be read or modified while the program is running:

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





---

# 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	Chapter Name	Page
13	Frequency Meter Principle	99
14	Frequency Meter Type	101



---

# Chapter 13

## Frequency Meter Principle

---

### Description

#### Overview

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

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

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

#### Operation Limits

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

The maximum duty cycle at 200 kHz is 60%.



---

# Chapter 14

## Frequency Meter Type

---

### Overview

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

### What Is in This Chapter?

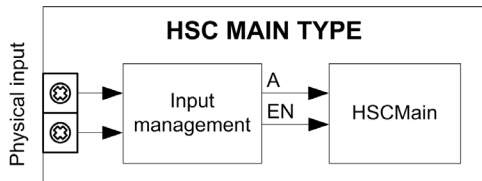
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	102
Configuration of the Frequency Meter Type	103
Programming	104

## 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 following function:

- Enable function ([see page 138](#))

## Configuration of the Frequency Meter Type

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>Frequency Meter</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <a href="#">see page 142</a> ).
5	Set the value of the <b>Range</b> → <b>Time base</b> parameter to determine the period during which the number of events is counted. Select the measurement of the update cycle time: <ul style="list-style-type: none"> <li>● 10 ms</li> <li>● 100 ms</li> <li>● 1000 ms (default value)</li> </ul>
6	Optionally, set the value of the <b>Control inputs</b> → <b>EN input</b> → <b>Location</b> parameter to enable the Enable Function ( <a href="#">see page 138</a> ).


## Programming

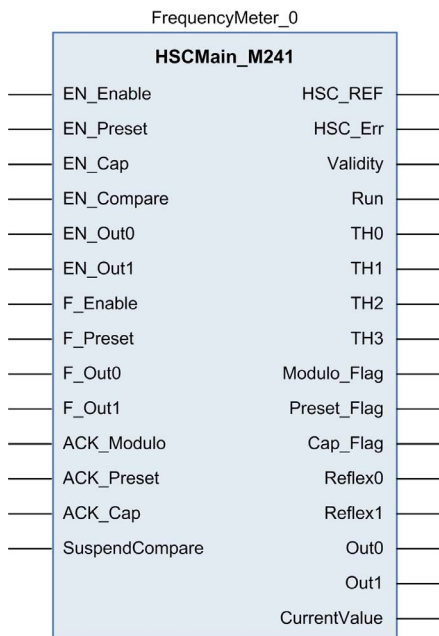
### Overview

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

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

### Adding the HSCMain Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>HSC</b> → <b>HSCMain_M241</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Type the <b>Main</b> 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: <b>Global Variables</b> → <b>&lt;MyController&gt;</b> → <b>PLC Logic</b> → <b>IoConfig_Globals</b> .





## 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	When <b>EN</b> input is configured: if <b>TRUE</b> , authorizes the counter enable via the Enable input ( <a href="#">see page 138</a> ).
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	<b>TRUE</b> = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, restarts the internal timer relative to the time base.
F_Out0	BOOL	Not used
F_Out1	BOOL	Not used
ACK_Modulo	BOOL	Not used
ACK_Preset	BOOL	On rising edge, resets .
ACK_Cap	BOOL	Not used
SuspendCompare	BOOL	Not used <b>NOTE:</b> EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF ( <a href="#">see page 152</a> )	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the <b>Administrative</b> function blocks.
HSC_Err	BOOL	<b>TRUE</b> = indicates that an error was detected. Use the EXPERTGetDiag ( <a href="#">see page 156</a> ) function block to get more information about this detected error.
Validity	BOOL	<b>TRUE</b> = indicates that output values on the function block are valid.
Run	BOOL	Not relevant
TH0	BOOL	Not relevant
TH1	BOOL	Not relevant
TH2	BOOL	Not relevant

## Frequency Meter Type

---

Outputs	Type	Comment
TH3	BOOL	Not relevant
Modulo_Flag	BOOL	Not relevant
Preset_Flag	BOOL	Set to 1 by the preset of the counter ( <i>see page 134</i> )
Cap_Flag	BOOL	Not relevant
Reflex0	BOOL	Not relevant
Reflex1	BOOL	Not relevant
Out0	BOOL	Not relevant
Out1	BOOL	Not relevant
CurrentValue	DINT	Current counter value of the counter.

---

# 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	Chapter Name	Page
15	Period Meter Type Principle	109
16	<b>Period Meter Type</b>	111



---

# Chapter 15

## Period Meter Type Principle

---

### Description

#### Overview

Use the **Period meter** type to:

- determine the duration of an event
- determine the time between 2 events
- set and measure the execution time for a process

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

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

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

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

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

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

**0.001 (1  $\mu$ s)** measure = 0.1 ms

**0.1 (100  $\mu$ s)** measure = 10 ms

**1 (1000  $\mu$ s)** measure = 100 ms

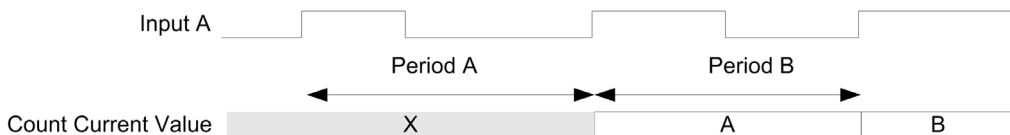
#### Edge to Opposite Mode

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



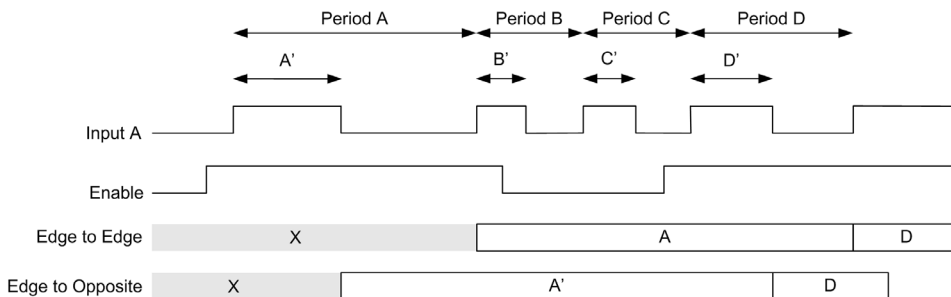
### Edge to Edge Mode

When the Enable condition = 1, the measurement is taken between 2 rising edges of the A input. The counting register is updated as soon as the second rising edge is detected.



### Enable Condition Interruption Behavior

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



### Operating Limits

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

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

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

---

# Chapter 16

## Period Meter Type

---

### Overview

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

### What Is in This Chapter?

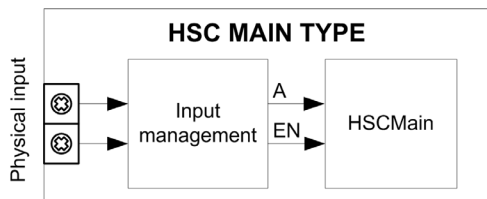
This chapter contains the following topics:

Topic	Page
Synopsis Diagram	112
Configuration of the Period Meter Type in Edge to Edge Mode	113
Configuration of the Period Meter Type in Edge to Opposite Mode	114
Programming	115
Adjusting Parameters	118

## 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 138](#))



## Configuration of the Period Meter Type in Edge to Edge Mode

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>Period Meter</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>General</b> → <b>PeriodMeter Mode</b> parameter to <b>Edge to Edge</b> .
5	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the inputs. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <i>see page 142</i> ).
6	Set the value of the <b>Range</b> → <b>Resolution</b> parameter. Select the unit of measurement: <ul style="list-style-type: none"> <li>● 1 <math>\mu\text{s}</math> (default value)</li> <li>● 100 <math>\mu\text{s}</math></li> <li>● 1000 <math>\mu\text{s}</math></li> </ul>
7	Enter the value of the <b>Range</b> → <b>Timeout</b> parameter to set the time value that a measured period shall not exceed.
8	Optionally, set the value of the <b>Control inputs</b> → <b>EN input</b> → <b>Location</b> parameter to enable the Enable Function ( <i>see page 138</i> ).

## Configuration of the Period Meter Type in Edge to Opposite Mode

### Procedure

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

Step	Action
1	Double-click <b>MyController</b> → <b>Counters</b> . <b>Result:</b> <b>Counters</b> editor tab opens for HSC configuration.
2	In the <b>Counters</b> editor tab, set the value of the <b>Counting function</b> parameter to <b>Period Meter</b> . <b>Result:</b> Depending on the selected counter function, the configuration parameters appear in the <b>Counters</b> editor tab.
3	If necessary, enter the value of the <b>General</b> → <b>Instance name</b> parameter. <b>NOTE:</b> <b>Instance name</b> is automatically given by the software and can be used as it is for the counter function block.
4	Set the value of the <b>General</b> → <b>PeriodMeter Mode</b> parameter to <b>Edge to Opposite</b> .
5	Set the value of the <b>Counting inputs</b> → <b>A input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the inputs. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <a href="#">see page 142</a> ).
6	Set the value of the <b>Range</b> → <b>Resolution</b> parameter. Select the unit of measurement: <ul style="list-style-type: none"> <li>● 1 <math>\mu\text{s}</math> (default value)</li> <li>● 100 <math>\mu\text{s}</math></li> <li>● 1000 <math>\mu\text{s}</math></li> </ul>
7	Enter the value of the <b>Range</b> → <b>Timeout</b> parameter to set the time value that a measured period shall not exceed.
8	Optionally, set the value of the <b>Control inputs</b> → <b>EN input</b> → <b>Location</b> parameter to enable the Enable Function ( <a href="#">see page 138</a> ).


## Programming

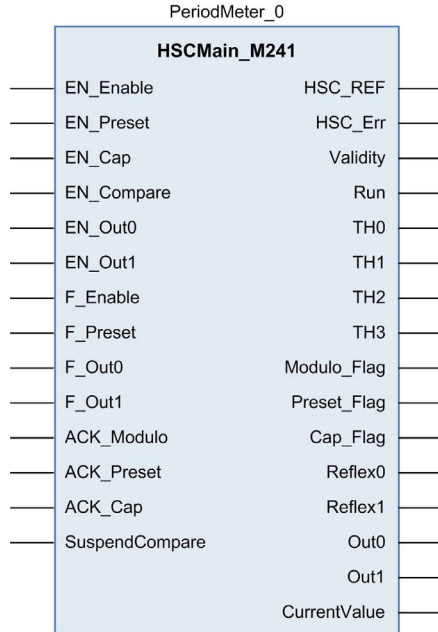
### Overview

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

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

### Adding the HSCMain Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>HSC</b> → <b>HSCMain_M241</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Type the <b>Main</b> 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: <b>Global Variables</b> → <b>&lt;MyController&gt;</b> → <b>PLC Logic</b> → <b>IoConfig_Globals</b> .



### 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 <b>EN</b> input is configured: if <b>TRUE</b> , authorizes the counter enable via the Enable input ( <i>see page 138</i> ).
EN_Preset	BOOL	Not used
EN_Cap	BOOL	Not used
EN_Compare	BOOL	Not used
EN_Out0	BOOL	Not used
EN_Out1	BOOL	Not used
F_Enable	BOOL	<b>TRUE</b> = authorizes changes to the current counter value.
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 <b>NOTE:</b> EN_Compare, EN_Reflex0, EN_Reflex1, F_Out0, F_Out1 remain operational while SuspendCompare is set.

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF ( <i>see page 152</i> )	Reference to the HSC. To be used with the EXPERT_REF_IN input pin of the <b>Administrative</b> function blocks.
HSC_Err	BOOL	<b>TRUE</b> = indicates that an error was detected. Use the EXPERTGetDiag ( <i>see page 156</i> ) function block used to get more information about this detected error.
Validity	BOOL	<b>TRUE</b> = indicates that output values on the function block are valid. If the time-out value is exceeded, Validity = <b>FALSE</b> .
Run	BOOL	<b>TRUE</b> = Counter is running.
TH0	BOOL	Not relevant
TH1	BOOL	Not relevant

---

<b>Outputs</b>	<b>Type</b>	<b>Comment</b>
TH2	BOOL	Not relevant
TH3	BOOL	Not relevant
Modulo_Flag	BOOL	Not relevant
Preset_Flag	BOOL	Not relevant
Cap_Flag	BOOL	This flag must be reset before a new capture can occur.
Reflex0	BOOL	Not relevant
Reflex1	BOOL	Not relevant
Out0	BOOL	Not relevant
Out1	BOOL	Not relevant
CurrentValue	DINT	Current 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 160) or `EXPERTSetParam` (see page 162) function blocks.

**NOTE:** Parameters set via the program override the parameters values configured in the HSC configuration window. Initial configuration parameters are restored on cold or warm start (see *Modicon M241 Logic Controller, Programming Guide*).

### Adjustable Parameters

This table provides the list of parameters from the `EXPERT_PARAMETER_TYPE` (see page 150) which can be read or modified while the program is running:

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

---

# Part VIII

## Optional Functions

---

### Overview

This part provides information on optional functions for HSC.

### What Is in This Part?

This part contains the following chapters:

Chapter	Chapter Name	Page
17	Comparison Function	121
18	Capture Function	129
19	Preset and Enable Functions	133





---

# Chapter 17

## Comparison Function

---

### Overview

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

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Comparison Principle with a <b>Main</b> type	122
Configuration of the Comparison on a <b>Main</b> Type	126
External Event Configuration	127

## Comparison Principle with a Main type

### Overview

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

- One-shot ([see page 35](#))
- Modulo-loop ([see page 49](#))
- Free-Large ([see page 69](#))

Comparison is configured in the Configuration screen ([see page 126](#)) by activating at least one threshold.

Comparison can be used to trigger:

- programming action on thresholds ([see page 123](#))
- an event on threshold associated with an external task ([see page 123](#))
- reflex outputs ([see page 124](#))

## Principle of a Comparison

The **Main** type can manage up to 4 thresholds.

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

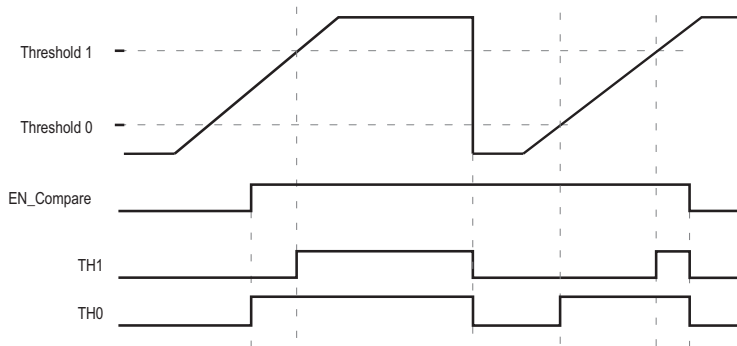
They are defined by configuration and can also be adjusted in the application program by using the `EXPERTSetParam` (see page 162) function block.

If `Thresholdx` ( $x = 0, 1, 2, 3$ ) is configured and comparison is enabled (`EN_Compare = 1`), output pin `THx` of the `HSCMain_M241` function block is:

- set when counter value  $\geq$  `Thresholdx`
- reset when counter value  $<$  `Thresholdx`

**NOTE:** When `EN_Compare` is set to 0 on `HSCMain_M241` function block, comparison functions are disabled, including external tasks triggered by a threshold event and Reflex outputs.

Example for 2 thresholds:



## Threshold Behavior

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

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

## Configuring Event

Configuring an event on threshold crossing allows to trigger an external task (see page 127). You can choose to trigger an event when a configured threshold is crossed downward, upward, or both.

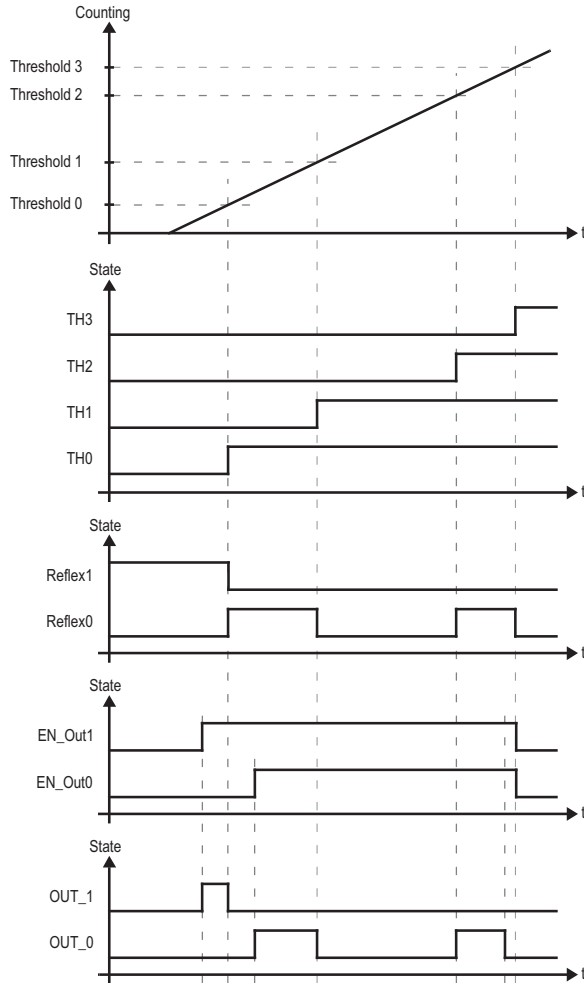
### Reflex Output Behavior

Configuring reflex outputs allows to trigger physical reflex outputs.

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

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

Example of the reflex outputs triggered by threshold:



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

## Changing the Threshold Values

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

### WARNING

#### UNINTENDED EQUIPMENT OPERATION

- Do not change the Threshold values without using the `SuspendCompare` input if `EN_Compare = 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:

Step	Action
1	<p>Set <code>SuspendCompare</code> to 1.</p> <p>The comparison is frozen at the current value:</p> <ul style="list-style-type: none"> <li>• <code>TH0</code>, <code>TH1</code>, <code>Reflex0</code>, <code>Reflex1</code>, <code>Out0</code>, <code>Out1</code> output bits of the block maintain their last value.</li> <li>• Physical Outputs 0, 1 maintain their last value</li> <li>• Events are masked</li> </ul> <p><b>NOTE:</b> <code>EN_Compare</code>, <code>EN_Out0</code>, <code>EN_Out1</code>, <code>F_Out0</code>, <code>F_Out1</code> remain operational while <code>SuspendCompare</code> is set.</p>
2	<p>Modify the Threshold values as needed using the <code>EXPERTSetParam</code> (see page 160) function block.</p> <p><b>NOTE:</b> Follow this rule to configure the threshold values: <code>TH0 &lt; TH1 &lt; TH2 &lt; TH3</code>.</p>
3	<p>Set <code>SuspendCompare</code> to 0.</p> <p>The new Threshold values are applied and the comparison is resumed.</p>

## Configuration of the Comparison on a Main Type

### Configuration Procedure

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

Step	Action
1	In the <b>Devices tree</b> , double-click <b>MyController</b> → <b>Counters</b> .
2	Set the value of the <b>Counting function</b> parameter to <b>HSC Main Single Phase</b> or <b>HSC Main Dual Phase</b> .
3	In the <b>Number of thresholds</b> parameter, select the number of thresholds to use.
4	Select the thresholds values. <b>NOTE:</b> Follow this rule to configure the threshold values: TH0 < TH1 < TH2 < TH3
5	Optionally, provide an event condition ( <i>see page 127</i> ).
6	Optionally, configure the <b>Reflex Output</b> behavior ( <i>see page 122</i> ).

## External Event Configuration

### Procedure

The following procedure describes how to configure an external event (see *Modicon M241 Logic Controller, Programming Guide*) to activate a task:

Step	Action
1	In the <b>Applications tree</b> tab, add a task.
2	Double-click the task node to associate to an external event.
3	In the <b>Type</b> dropdown menu, select <b>External</b> .
4	In the <b>External Event</b> 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
I0	Task is activated when the input I0 is set to 1.
I1	Task is activated when the input I1 is set to 1.
I2	Task is activated when the input I2 is set to 1.
I3	Task is activated when the input I3 is set to 1.
I4	Task is activated when the input I4 is set to 1.
I5	Task is activated when the input I5 is set to 1.
I6	Task is activated when the input I6 is set to 1.
I7	Task is activated when the input I7 is set to 1.
HSC0_TH0	Task is activated when the threshold TH0 of the HSC0 is set to 1.
HSC0_TH1	Task is activated when the threshold TH1 of the HSC0 is set to 1.
HSC0_TH2	Task is activated when the threshold TH2 of the HSC0 is set to 1.
HSC0_TH3	Task is activated when the threshold TH3 of the HSC0 is set to 1.
HSC0_Stop	Task is activated when the HSC0 is set to 0.
HSC1_TH0	Task is activated when the threshold TH0 of the HSC1 is set to 1.
HSC1_TH1	Task is activated when the threshold TH1 of the HSC1 is set to 1.
HSC1_TH2	Task is activated when the threshold TH2 of the HSC1 is set to 1.
HSC1_TH3	Task is activated when the threshold TH3 of the HSC1 is set to 1.
HSC1_Stop	Task is activated when the HSC1 is set to 0.





---

# Chapter 18

## Capture Function

---

### Overview

This chapter provides information on capture function for HSC.

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Capture Principle with a <b>Main</b> Type	130
Configuration of the Capture on a <b>Main</b> Type	131

## Capture Principle with a Main Type

### Overview

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

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

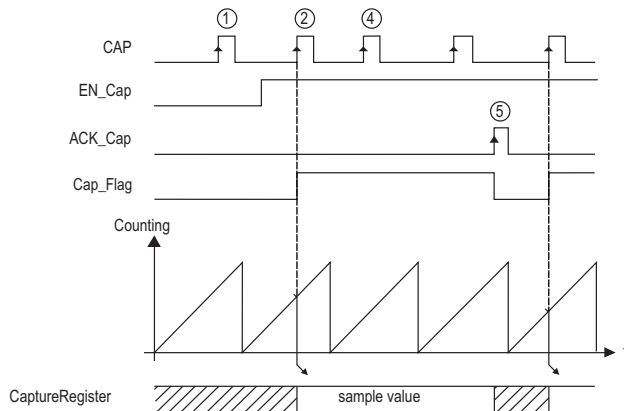
- One-shot ([see page 41](#))
- Modulo-loop ([see page 61](#))
- Free-large ([see page 77](#))

Using this function requires to:

- configure the optional Capture input: **CAP**
- use `EXPERTGetCapturedValue` ([see page 154](#)) 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 <b>EN_Cap</b> = 0, the function is not operational.
2	When <b>EN_Cap</b> = 1, the edge on <b>CAP</b> captures the current counter value and puts it into the Capture register, and triggers the rising edge of <b>Cap_Flag</b> .
3	Get the stored value using <code>EXPERTGetCapturedValue</code> ( <a href="#">see page 154</a> ).
4	While <b>Cap_Flag</b> = 1, any new edge on the physical input <b>CAP</b> is ignored.
5	The rising edge of <code>HSCMain_M241</code> ( <a href="#">see page 164</a> ) function block input <b>ACK_Cap</b> triggers the falling edge <b>Cap_Flag</b> 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 <b>Devices tree</b> , double-click <b>MyController</b> → <b>Counters</b> .
2	Set the value of the <b>Counting function</b> parameter to <b>HSC Main Single Phase</b> or <b>HSC Main Dual Phase</b> .
3	Select a value for the <b>Capture</b> → <b>CAP input</b> → <b>Location</b> parameter to <b>I3</b> or <b>I10</b> .
4	Select a value for the <b>Capture</b> → <b>CAP input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table ( <a href="#">see page 142</a> ).
5	Select a triggering mode for the <b>Capture</b> → <b>Mode</b> parameter: <ul style="list-style-type: none"> <li>● Preset (<a href="#">see page 134</a>) (default value)</li> <li>● <b>CAP Rising</b></li> <li>● <b>CAP Falling</b></li> <li>● <b>CAP Both</b></li> </ul>



---

# Chapter 19

## Preset and Enable Functions

---

### Overview

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

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Preset Function	134
Free-large or Period Meter Preset Conditions	136
Enable Function	138

## Preset Function

### Overview

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

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

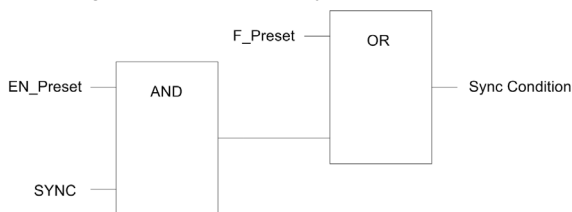
- **One shot** counter: preset and start the counter
- **Modulo-loop** counter: reset and start the counter
- **Event counting**: restart the internal time base at the beginning

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

### Description

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

This diagram illustrates the Sync conditions of the HSC:



**EN\_Preset** input of the HSC function block

**F\_Preset** input of the HSC function block

**SYNC** physical input SYNC

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

The Sync condition operates on a rising edge.

### Configuration

This procedure describes how to configure a preset function:

Step	Action
1	In the <b>Devices tree</b> , double-click <b>MyController</b> → <b>Counters</b> .
2	Set the value of the <b>Counting function</b> parameter to <b>HSC Main Single Phase</b> or <b>HSC Main Dual Phase</b> .
3	Select the value of the <b>Control inputs</b> → <b>SYNC input</b> → <b>Location</b> parameter.
4	Select the value of the <b>Control inputs</b> → <b>SYNC input</b> → <b>Bounce filter</b> parameter.
5	Select the value of the <b>Control inputs</b> → <b>Edge detection</b> 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
<b>Sync Rising</b>	Detection done on rising edge.
<b>Sync Falling</b>	Detection done on falling edge.
<b>Sync Both</b>	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 create by using 2 inputs:

- SYNC
- REF

There are preset conditions available:

- At the edge of the input SYNC (rising)
- 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)

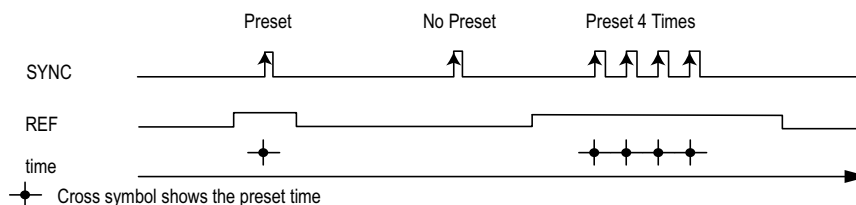
The counter synchronizes upon the encoder reference point.

### At the Rising Edge of the Input REF

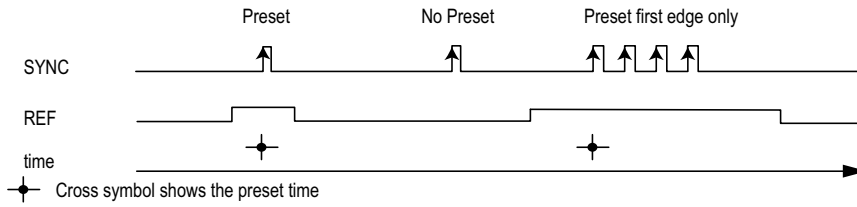
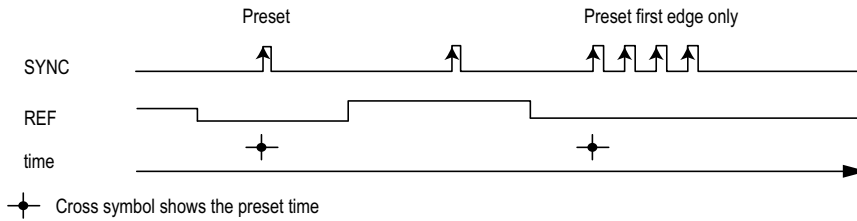
The counter synchronizes upon the mechanical position.

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

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





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

## Enable Function

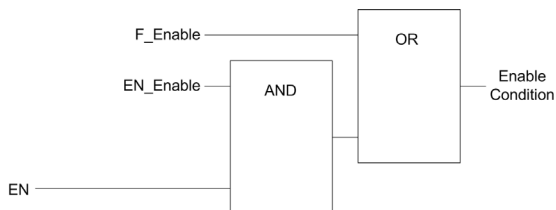
### Overview

The enable function is used to authorize the counting operation.

### Description

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

The diagrams illustrates the enable conditions:



**EN\_Enable** input of the HSC function block

**F\_Enable** input of the HSC function block

**EN** physical input Enable

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

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

### Configuration

This procedure describes how to configure an enable function:

Step	Action
1	In the <b>Devices tree</b> , double-click <b>MyController</b> → <b>Counters</b> .
2	Select the <b>Counters</b> tab.
3	Set the value of the <b>Control inputs</b> → <b>EN input</b> → <b>Location</b> parameter to <b>I1</b> .
4	Select the value of the <b>Control inputs</b> → <b>EN input</b> → <b>Bounce filter</b> parameter to reduce the bounce effect on the input. The filtering value determines the counter maximum frequency as shown in the Bounce Filter table (see page 142).

---

# Appendices

---



## Overview

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

## What Is in This Appendix?

The appendix contains the following chapters:

Chapter	Chapter Name	Page
A	General Information	141
B	Data Types	145
C	Function Blocks	153
D	Function and Function Block Representation	169



---

# Appendix A

## General Information

---

### Overview

The information described in this chapter is common for FG/PTO/PWM and HSC features.

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Dedicated Features	142
General Information on Administrative and Motion Function Block Management	143

## 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
A	0.000	200 kHz
B	0.001	200 kHz
EN	0.002	200 kHz
CAP	0.005	100 kHz
	0.010	50 kHz
	0.05	25 kHz
	0.1	5 kHz
	0.5	1 kHz
	1	500 Hz
	5	100 Hz

A is the counting input of the counter.  
 B is the counting input of the dual phase counter.  
 EN is the enable input of the counter.  
 CAP is the capture input of the counter.

### Dedicated Outputs

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

## WARNING

### UNINTENDED EQUIPMENT OPERATION

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

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

---

## General Information on Administrative and Motion Function Block Management

### Management of Input Variables

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

Any further modifications of the input variables are not taken 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:

Topic	Page
EXPERT_ERR_TYPE: Type for Error Variable of EXPERT Function Block	146
EXPERT_FREQMETER_TIMEBASE_TYPE: Type for Frequency Meter Time Base Variable	147
EXPERT_HSCMAIN_TIMEBASE_TYPE: Type for HSC Main Time Base Variable	148
EXPERT_IMMEDIATE_ERR_TYPE: Type for Error Variable of the GetImmediateValue Function Block	149
EXPERT_PARAMETER_TYPE: Type for Parameters to Get or to Set on EXPERT	150
EXPERT_PERIODMETER_RESOLUTION_TYPE: Type for Period Meter Time Base Variable	151
EXPERT_REF: EXPERT Reference Value	152

## 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 <code>PARAMETER_TYPE</code> section for valid parameters ( <i>see page 150</i> ).
EXPERT_INVALID_PARAMETER	03 hex	The value of the parameter is incorrect. For example, <code>Preset Value</code> is <code>&lt;TH1</code> or <code>&lt;TH0</code> .
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\_HSCMAIN\_TIMEBASE\_TYPE: Type for HSC Main Time Base Variable****Enumerated Type Description**

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

<b>Name</b>	<b>Value</b>
EXPERT_HSCMAIN_100ms	00 hex
EXPERT_HSCMAIN_1s	01 hex
EXPERT_HSCMAIN_10s	02 hex
EXPERT_HSCMAIN_60s	03 hex

## 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_FUNC_NO_ERROR	00 hex	No error detected
EXPERT_IMMEDIATE_FUNC_UNKNOWN	01 hex	The reference of IMMEDIATE function is incorrect or not configured
EXPERT_IMMEDIATE_FUNC_UNKNOWN_PARAMETER	02 hex	A parameter reference is incorrect

## EXPERT\_PARAMETER\_TYPE: Type for Parameters to Get or to Set on EXPERT

### 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 value of an EXPERT function.
EXPERT_MODULO	01 hex	To get or set the Modulo value of an EXPERT function.
EXPERT_TIMEBASE	03 hex	To get or set the Timebase value ( <a href="#">see page 148</a> ) of an EXPERT function.
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\_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_1 $\mu$ s	00 hex
EXPERT_PERIODMETER_100 $\mu$ s	01 hex
EXPERT_PERIODMETER_1000 $\mu$ s	02 hex

## **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.



---

# Appendix C

## Function Blocks

---

### Overview

This chapter describes the functions and the function blocks of the HSC Library.

### What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
EXPERTGetCapturedValue: Returns Content of Capture Registers	154
EXPERTGetDiag: Provides Detail of Detected Error on a Principal EXPERT I/O Function	156
EXPERTGetImmediateValue: Read Counter Value of HSC	158
EXPERTGetParam: Returns Parameters of Principal EXPERT I/O Function	160
EXPERTSetParam: Adjust Parameters of a HSC	162
HSCMain_M241: HSC Main Function Block	164
HSCSimple_M241: HSC Simple Function Block	167

## EXPERTGetCapturedValue: Returns Content of Capture Registers

### Function Description

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

### Graphical Representation



### IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* (see page 169).

### I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF (see page 152)	Reference to the EXPERT function block. Must not be changed during block execution.
Execute	BOOL	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
CaptureNumber	BYTE	Index of the capture register: <ul style="list-style-type: none"> <li>● for HSCMain: always 0</li> <li>● for Cap0: 0</li> <li>● for Cap1: 1</li> </ul>

This table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF (see page 152)	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 146)	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 143).

### Adding the EXPERTGetCapturedValue Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>Administrative</b> → <b>EXPERTGetCapturedValue</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Link the <b>EXPERT_REF_IN</b> input to the <b>HSC_REF</b> output of the HSC.

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

### Function Description

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

### Graphical Representation



### IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* (see page 169).

### I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF (see page 152)	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 152)	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 146)	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 143).

This table indicates the diagnostic values:

Bit	BASE (HSCMain or HSCSimple)	Description
0	–	Inputs error detected
1	–	Outputs error detected
2	–	–
3	–	–
4	–	Internal error detected on self test
5	–	–
6	–	Bus error detected
7	Configuration error detected	Configuration error detected
8	–	–
9	–	Configuration initialization
10	–	Sensor supply low voltage
11	–	Actuator supply low voltage
12	–	Output0 short-circuit or over-current detected
13	–	Output1 short-circuit or over-current detected
14	–	–
15	–	–

### Adding the EXPERTGetDiag Function Block

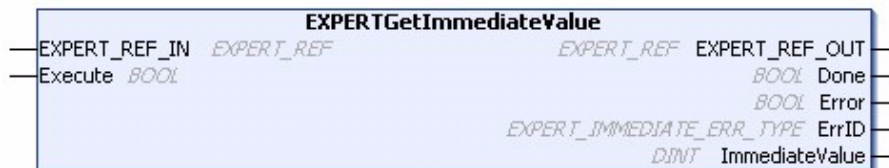
Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>Administrative</b> → <b>EXPERTGetDiag</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Link the <b>EXPERT_REF_IN</b> input to the <b>HSC_REF</b> output of the HSC.

## EXPERTGetImmediateValue: Read Counter Value of HSC

### Function Description

This administrative function permits to read the counter value of an HSC 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 169).

### I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF (see page 152)	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 152)	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 149)	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 <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>Administrative</b> → <b>EXPERTGetImmediateValue</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Link the <b>EXPERT_REF_IN</b> input to the <b>HSC_REF</b> output of the HSC.

## EXPERTGetParam: Returns Parameters of Principal EXPERT I/O Function

### Function Description

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

### Graphical Representation



### IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* (see page 169).

### I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF (see page 152)	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 150)	Parameter to read.



This table describes the output variables:

Outputs	Type	Comment
EXPERT_REF_OUT	EXPERT_REF (see page 152)	Reference to the EXPERT function block.
Done	BOOL	TRUE = indicates that ParamValue is valid. Function block execution is finished.
Busy	BOOL	TRUE = indicates that the function block execution is in progress.
Error	BOOL	TRUE = indicates that an error was detected. Function block execution is finished.
ErrID	EXPERT_ERR_TYPE (see page 146)	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 143).

### Adding the EXPERTGetParam Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>Administrative</b> → <b>EXPERTGetParam</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Link the <b>EXPERT_REF_IN</b> input to the <b>HSC_REF</b> output of the HSC.

## EXPERTSetParam: Adjust Parameters of a HSC

### Function Description

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

### Graphical Representation



### IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* (see page 169).

### I/O Variables Description

This table describes the input variables:

Inputs	Type	Comment
EXPERT_REF_IN	EXPERT_REF (see page 152)	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 150)	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 152)	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 146)	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 143).

### Adding the EXPERTSetParam Function Block

Step	Description
1	Select the <b>Libraries</b> tab in the <b>Software Catalog</b> and click <b>Libraries</b> . Select <b>Controller</b> → <b>M241</b> → <b>M241 HSC</b> → <b>Administrative</b> → <b>EXPERTSetParam</b> in the list, drag-and-drop the item onto the <b>POU</b> window.
2	Link the <b>EXPERT_REF_IN</b> input to the <b>HSC_REF</b> output of the HSC.

## HSCMain\_M241: HSC Main Function Block

### Function Description

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

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

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

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

### ⚠ WARNING

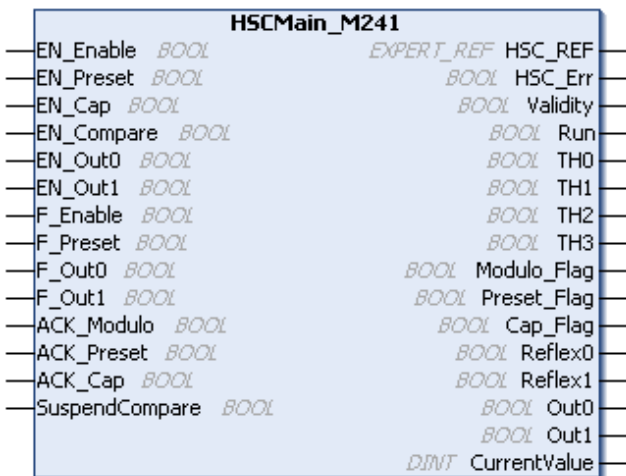
#### UNINTENDED OUTPUT VALUES

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

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

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

### Graphical Representation



## IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* (see page 169).

## I/O Variables Description

This table describes the input variables:

Input	Type	Description
EN_Enable	BOOL	TRUE = enables the counter.
EN_Preset	BOOL	TRUE = authorizes the counter synchronization and start via the Sync input.
EN_Cap	BOOL	TRUE = enables the Capture input (if configured in <b>One shot</b> , <b>Modulo loop</b> , <b>Free large modes</b> ).
EN_Compare	BOOL	TRUE = enables the comparator operation (using Thresholds 0, 1, 2, 3): <ul style="list-style-type: none"> <li>● basic comparison (TH0, TH1, TH2, TH3 output bits)</li> <li>● reflex (Reflex0, Reflex1 output bits)</li> <li>● events (to trigger external tasks on threshold crossing)</li> </ul>
EN_Out0	BOOL	TRUE = enables Output0 to echo the Reflex0 value (if configured in <b>One shot</b> , <b>Modulo loop</b> , <b>Free large modes</b> ).
EN_Out1	BOOL	TRUE = enables Output1 to echo the Reflex1 value (if configured in <b>One shot</b> , <b>Modulo loop</b> , <b>Free large modes</b> ).
F_Enable	BOOL	TRUE = authorizes changes to the current counter value.
F_Preset	BOOL	On rising edge, authorizes counting function synchronization and start in the following counting modes: <b>One-shot counter:</b> to preset and start the counter <b>Modulo loop counter:</b> to reset and start the counter <b>Free large counter:</b> to preset and start the counter <b>Event counter:</b> to restart the internal time base at the beginning <b>Frequency meter:</b> to restart the internal time base at the beginning
F_Out0	BOOL	TRUE = forces Output0 to 1 (if configured in <b>One-shot</b> , <b>Modulo loop</b> , <b>Free large modes</b> ).
F_Out1	BOOL	TRUE = forces Output1 to TRUE (if configured in <b>One-shot</b> , <b>Modulo loop</b> , <b>Free large modes</b> ).
ACK_Modulo	BOOL	On rising edge, resets Modulo_Flag ( <b>Modulo loop</b> and <b>Free large modes</b> ).
ACK_Preset	BOOL	On rising edge, resets Preset_Flag.
ACK_Cap	BOOL	On rising edge, resets the Cap_Flag ( <b>One-shot</b> , <b>Modulo loop</b> , <b>Free large modes</b> ).
SuspendCompare	BOOL	TRUE = compare results are suspended: <ul style="list-style-type: none"> <li>● TH0, TH1, TH2, TH3, Reflex0, Reflex1, Out0, Out1 output bits of the block maintain their last value.</li> <li>● Physical outputs Output0 and Output1 maintain their last value.</li> <li>● Events are masked.</li> </ul> <b>NOTE:</b> 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 152)	Reference to the HSC.
Validity	BOOL	TRUE = indicates that output values on the function block are valid. In the <b>Period Meter Type</b> , if the time-out value is exceeded, Validity = FALSE.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the HSCGetDiag (see page 156) function block to get more information about this detected error.
Run	BOOL	TRUE = counter is running. In <b>One-shot mode</b> , the Run bit switches to 0 when CurrentValue reaches 0.
TH0	BOOL	TRUE = current counter value > Threshold 0 (if configured in <b>One shot, Modulo loop, Free large modes</b> ). Only active when EN_Compare is set.
TH1	BOOL	TRUE = current counter value > Threshold 1 (if configured in <b>One shot, Modulo loop, Free large modes</b> ). Only active when EN_Compare is set.
TH2	BOOL	TRUE = current counter value > Threshold 2 (if configured in <b>One-shot, Modulo loop, Free large modes</b> ). Only active when EN_Compare is set.
TH3	BOOL	TRUE = current counter value > Threshold 3 (if configured in <b>One-shot, Modulo loop, Free large modes</b> ). Only active when EN_Compare is set.
Modulo_Flag	BOOL	Set to TRUE by the rolls over of: <ul style="list-style-type: none"> <li>● <b>Modulo loop counter</b>: when the counter rolls over to the modulo or 0</li> <li>● <b>Free large counter</b>: when the counter roll overs its limits</li> </ul>
Preset_Flag	BOOL	Set to TRUE by the synchronization of: <ul style="list-style-type: none"> <li>● <b>One-shot counter</b>: when the counter presets and starts</li> <li>● <b>Modulo loop counter</b>: when the counter resets</li> <li>● <b>Free large counter</b>: when the counter presets</li> <li>● <b>Event counter</b>: when the internal timer relative to the time base restarts</li> <li>● <b>Frequency meter</b>: when the internal timer relative to the time base restarts</li> </ul>
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 <b>One shot, Modulo loop, Free large modes</b> ). Only active when EN_Compare is set.
Reflex1	BOOL	State of Reflex1 (if configured in <b>One shot, Modulo loop, Free large modes</b> ). Only active when EN_Compare is set.
Out0	BOOL	Indicates the state of Output0.
Out1	BOOL	Indicates the state of Output1.
CurrentValue	DINT	Current counter value of the counter.

## HSCSimple\_M241: HSC Simple Function Block

### Function Description

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

- one-way counting
- no threshold

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

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

### ⚠ WARNING

#### UNINTENDED OUTPUT VALUES

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

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

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

### Graphical Representation



### IL and ST Representation

To see the general representation in IL or ST language, refer to *Function and Function Block Representation* (see page 169).

## I/O Variables Description

This table describes the input variables:

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

This table describes the output variables:

Outputs	Type	Comment
HSC_REF	EXPERT_REF (see page 152)	Reference to the HSC.
HSC_Err	BOOL	TRUE = indicates that an error was detected. Use the <code>EXPERTGetDiag</code> (see page 156) function block used to get more information about this detected error.
Validity	BOOL	TRUE = indicates that the output values on the function block are valid.
Run	BOOL	TRUE = counter is running. In One-shot mode, switches to 0 when <code>CurrentValue</code> reaches 0. A rising edge on <code>Sync</code> is needed to restart the counter.
Modulo_Flag	BOOL	Set to 1 by the rolls over of a Modulo loop counter when the counter rolls over the modulo.
CurrentValue	DWORD	Current count value of the counter.



---

# 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:

Topic	Page
Differences Between a Function and a Function Block	170
How to Use a Function or a Function Block in IL Language	171
How to Use a Function or a Function Block in ST Language	175

## Differences Between a Function and a Function Block

### Function

A function:

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

**Examples:** boolean operators (AND), calculations, conversion (BYTE\_TO\_INT)

### Function Block

A function block:

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

**Examples:** timers, counters

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

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

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

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

### General Information

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

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

### Using a Function in IL Language

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

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

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

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

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

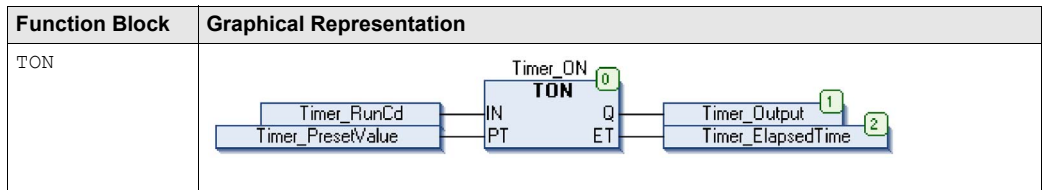
Function	Representation in SoMachine POU IL Editor															
IL example of a function without input parameter: IsFirstMastCycle	<pre> 1  PROGRAM MyProgram_IL 2  VAR 3      FirstCycle: BOOL; 4  END_VAR                     </pre> <hr/> <table border="1" data-bbox="371 459 979 570"> <tr> <td data-bbox="371 459 440 488">1</td> <td data-bbox="440 459 742 488"><b>IsFirstMastCycle</b></td> <td data-bbox="742 459 979 488"></td> </tr> <tr> <td data-bbox="371 488 440 521"></td> <td data-bbox="440 488 742 521"><b>ST</b></td> <td data-bbox="742 488 979 521">FirstCycle</td> </tr> <tr> <td data-bbox="371 521 440 553"></td> <td data-bbox="440 521 742 553"></td> <td data-bbox="742 521 979 553"></td> </tr> </table>	1	<b>IsFirstMastCycle</b>			<b>ST</b>	FirstCycle									
1	<b>IsFirstMastCycle</b>															
	<b>ST</b>	FirstCycle														
IL example of a function with input parameters: SetRTCDrift	<pre> 1  PROGRAM MyProgram_IL 2  VAR 3      myDrift: SINT (-29..29) := 5; 4      myDay: DAY_OF_WEEK := SUNDAY; 5      myHour: HOUR := 12; 6      myMinute: MINUTE; 7      myDiag: RTCSETDRIFT_ERROR; 8  END_VAR                     </pre> <hr/> <table border="1" data-bbox="371 971 930 1149"> <tr> <td data-bbox="371 971 440 1003">1</td> <td data-bbox="440 971 673 1003"><b>LD</b></td> <td data-bbox="673 971 930 1003">myDrift</td> </tr> <tr> <td data-bbox="371 1003 440 1036"></td> <td data-bbox="440 1003 673 1036"><b>SetRTCDrift</b></td> <td data-bbox="673 1003 930 1036">myDay</td> </tr> <tr> <td data-bbox="371 1036 440 1068"></td> <td data-bbox="440 1036 673 1068"></td> <td data-bbox="673 1036 930 1068">myHour</td> </tr> <tr> <td data-bbox="371 1068 440 1101"></td> <td data-bbox="440 1068 673 1101"></td> <td data-bbox="673 1068 930 1101">myMinute</td> </tr> <tr> <td data-bbox="371 1101 440 1133"></td> <td data-bbox="440 1101 673 1133"><b>ST</b></td> <td data-bbox="673 1101 930 1133">myDiag</td> </tr> </table>	1	<b>LD</b>	myDrift		<b>SetRTCDrift</b>	myDay			myHour			myMinute		<b>ST</b>	myDiag
1	<b>LD</b>	myDrift														
	<b>SetRTCDrift</b>	myDay														
		myHour														
		myMinute														
	<b>ST</b>	myDiag														

## Using a Function Block in IL Language

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

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

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



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

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

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

### General Information

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

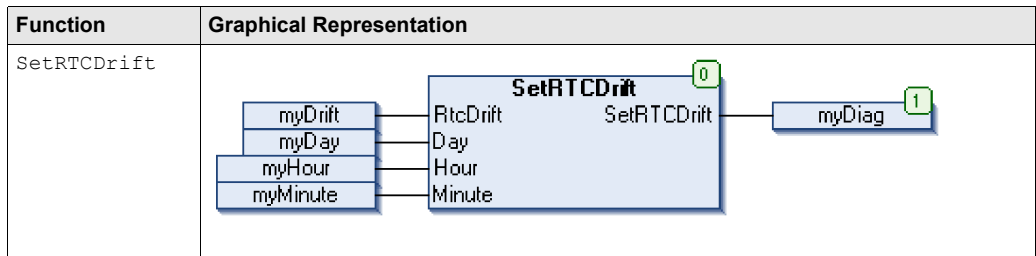
Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

### Using a Function in ST Language

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

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

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



The ST language of this function is the following:

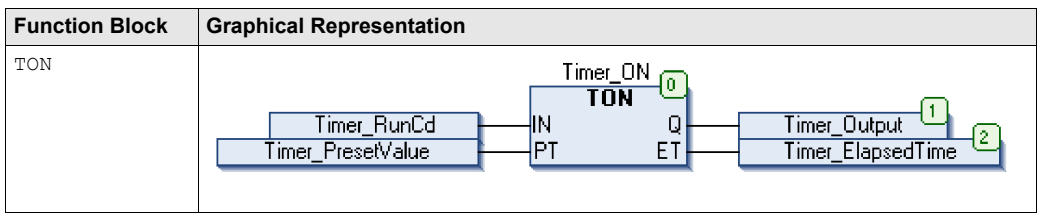
Function	Representation in SoMachine POU ST Editor
SetRTCDrift	<pre>PROGRAM MyProgram_ST VAR myDrift: SINT(-29..29) := 5; myDay: DAY_OF_WEEK := SUNDAY; myHour: HOUR := 12; myMinute: MINUTE; myRTCadjust: RTCDRIFT_ERROR; END_VAR  myRTCadjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);</pre>

### Using a Function Block in ST Language

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

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





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

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





## A

### **application**

A program including configuration data, symbols, and documentation.

## B

### **byte**

A type that is encoded in an 8-bit format, ranging from 16#00 to 16#FF in hexadecimal representation.

## C

### **CFC**

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

## F

### **FB**

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

### **function block diagram**

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

## H

### **HSC**

*(high-speed counter)*

## I

### ID

*(identifier/identification)*

### IEC 61131-3

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

### IL

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

### INT

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

## L

### LD

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

## N

### node

An addressable device on a communication network.

## P

### POU

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

### program

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

### PTO

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

**S****ST**

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

**V****variable**

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





## A

- adjusting functions
  - EXPERTGetParam, 160
  - EXPERTSetParam, 162

## B

- Busy
  - management of status variables, 143

## C

- Capture
  - HSCMain, 130
- CommandAborted
  - management of status variables, 143
- Comparison
  - HSCMain, 122
- counter blocks
  - HSCMain, 164
  - HSCSimple, 167

## D

- data types
  - EXPERT\_ERR\_TYPE, 146
  - EXPERT\_FREQMETER\_TIMEBASE\_TYPE, 147
  - EXPERT\_HSCMAIN\_TIMEBASE\_TYPE, 148
  - EXPERT\_IMMEDIATE\_ERR\_TYPE, 149
  - EXPERT\_PARAMETER\_TYPE, 150
  - EXPERT\_PERIODMETER\_RESOLUTION\_TYPE, 151
  - HSC\_REF, 152
- dedicated features, 142
- diagnostic functions
  - EXPERTGetDiag, 156
- Done
  - management of status variables, 143

## E

- Enable
  - Function, 138
- ErrID
  - handling a detected error, 143
  - management of status variables, 143
- Error
  - handling a detected error, 143
  - management of status variables, 143
- Event Counting
  - HSC Modes of Embedded HSC, 87
- Execute
  - management of status variables, 143
- EXPERT\_ERR\_TYPE, 146
- EXPERT\_FREQMETER\_TIMEBASE\_TYPE
  - data types, 147
- EXPERT\_HSCMAIN\_TIMEBASE\_TYPE
  - data types, 148
- EXPERT\_IMMEDIATE\_ERR\_TYPE, 149
- EXPERT\_PARAMETER\_TYPE, 150
- EXPERT\_PERIODMETER\_RESOLUTION\_TYPE
  - data types, 151
- EXPERTGetCapturedValue, 154
- EXPERTGetDiag, 156
- EXPERTGetImmediateValue, 158
- EXPERTGetParam, 160
- EXPERTSetParam, 162

## F

- Free-large
  - HSC Modes of Embedded HSC, 72
- frequency meter
  - description, 99
  - programming, 104
  - synopsis, 102
- Function
  - Enable, 138

function blocks

- EXPERTGetCapturedValue, 154
- EXPERTGetDiag, 156
- EXPERTGetImmediateValue, 158
- EXPERTGetParam, 160
- EXPERTSetParam, 162
- HSCMain, 164
- HSCSimple, 167

functions

- differences between a function and a function block, 170
- how to use a function or a function block in IL language, 171
- how to use a function or a function block in ST language, 175

## H

handling a detected error

- ErrID, 143
- Error, 143

HSC Modes of Embedded HSC

- Event Counting, 87
- Free-large, 72
- Modulo-loop, 51

HSC\_REF, 152

HSCMain

- Capture, 130
- Comparison, 122
- function blocks, 164

HSCSimple

- function blocks, 167

## M

management of status variables

- Busy, 143
- CommandAborted, 143
- Done, 143
- ErrID, 143
- Error, 143
- Execute, 143

Modulo-loop

- HSC Modes of Embedded HSC, 51

## P

period meter

- description, 109
- parameters, 118
- programming, 115
- synopsis, 112