

Modicon TM5

Strain Gauge IoDrvTM5SEAISG

Library Guide

09/2020

EIO0000003185.01

www.schneider-electric.com

Schneider
 Electric™

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

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

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

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

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

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

© 2020 Schneider Electric. All rights reserved.

Table of Contents



	Safety Information	5
	About the Book	7
Chapter 1	General Overview	11
	Create your Measurement System	11
Chapter 2	Programming	13
2.1	Strain Gauge Function Block	14
	Adding the StrainGaugeExt Function Block	15
	StrainGaugeExt Function Block Presentation	16
2.2	Strain Gauge Legacy Function Block	19
	Adding the StrainGauge Function Block	20
	StrainGauge Function Block Presentation	21
2.3	Calibrate Your System	23
	Linear Calibration	24
	Create the First Reference Point	25
	Create the Second Reference Point	26
	Taring Your System	27
2.4	Measure a Calibrated Value	28
	Using Your System	28
Appendices	29
Appendix A	Data Types	31
	StainGauge_Error: Error Codes	32
	StrainGaugeParameter: Calibration Parameters	33
Appendix B	Function and Function Block Representation	35
	Differences Between a Function and a Function Block	36
	How to Use a Function or a Function Block in IL Language	37
	How to Use a Function or a Function Block in ST Language	41
Glossary	45
Index	47

Safety Information



Important Information

NOTICE

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



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



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

DANGER

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

WARNING

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

CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book



At a Glance

Document Scope

This documentation will acquaint you with the strain gauge configuration and functions offered within the full-bridge strain gauge electronic module.

This documentation describes the function block and variables of the IoDrvTM5SEAI5G strain gauge library.

In order to use this manual, you must:

- Have a thorough understanding of the TM5SEAI5G, 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 for the release of EcoStruxure™ Machine Expert V1.2.5.

Related Documents

Title of Documentation	Reference Number
Modicon M258 Logic Controller Programming Guide	EIO0000004135 (Eng); EIO0000004136 (Fre); EIO0000004137 (Ger); EIO0000004138 (Spa); EIO0000004139 (Ita); EIO0000004140 (Chs)
Modicon LMC058 Motion Controller Programming Guide	EIO0000004165 (Eng); EIO0000004166 (Fre); EIO0000004167 (Ger); EIO0000004168 (Spa); EIO0000004169 (Ita); EIO0000004170 (Chs)

Title of Documentation	Reference Number
Modicon TM5 Expansion Modules Configuration Programming Guide	EIO0000003179 (Eng) , EIO0000003180 (Fre) , EIO0000003181 (Ger) , EIO0000003182 (Spa) , EIO0000003183 (Ita) , EIO0000003184 (Chs)
Modicon TM5 Analog I/O Modules Hardware Guide	EIO0000003203 (Eng) , EIO0000003204 (Fre) , EIO0000003205 (Ger) , EIO0000003206 (Spa) , EIO0000003207 (Ita) , EIO0000003208 (Chs)

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

Product Related Information

WARNING

LOSS OF CONTROL

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

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

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

WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Chapter 1

General Overview

Create your Measurement System

Overview

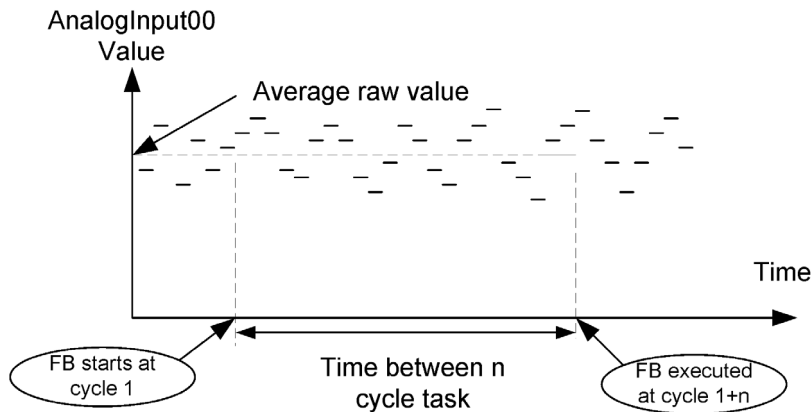
The full-bridge strain gauge sensor provides an electrical signal that the TM5SEASIG module converts to a raw numerical value. The raw value is then processed with the use of the StrainGageExt function block (*see page 16*) which provides a calibrated value.

This function block has 3 functions:

- make an average measure of the TM5SEASIG input in a defined period
- define a linear calibration to match the needs of your process
- provide a calibrated mesure

NOTE: The Strain Gauge function block is not automatically declared when you add TM5SEASIG to the controller.

The average raw value is calculated with all the measures done by the TM5SEASIG module during a define number of task cycles. The number of task cycles is set with the **Cycle_number** input of the function block.

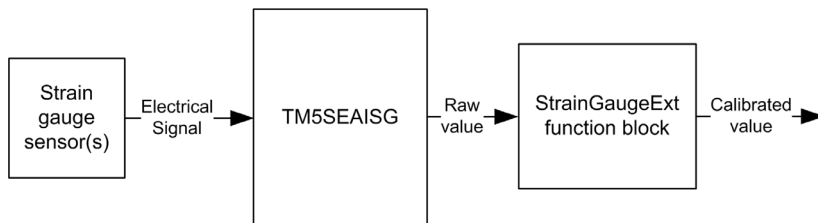


Where **n** is the **Cycle_number** value.

NOTE: The accuracy of the measurement provided by the electronic module may be compromised considerably if the full-bridge strain gauge set-up and installation rules have not been observed.

Measurement Cycle

The measurement system is shown in the following diagram:



Your measurement system is created through the configuration of the TM5SEAISG and the usage of the StrainGaugeExt function block.

Chapter 2

Programming

Overview

This chapter describes how to use the **StrainGaugeExt** function block to calibrate your measurement system and how to obtain a calibrated value.

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
2.1	Strain Gauge Function Block	14
2.2	Strain Gauge Legacy Function Block	19
2.3	Calibrate Your System	23
2.4	Measure a Calibrated Value	28

Section 2.1

Strain Gauge Function Block

What Is in This Section?


This section contains the following topics:

Topic	Page
Adding the StrainGaugeExt Function Block	15
StrainGaugeExt Function Block Presentation	16

Adding the StrainGaugeExt Function Block

Procedure

Follow these steps to add and create an instance of a **StrainGaugeExt** function block:

Step	Action
1	Select the Libraries tab in the Software Catalog and click Libraries . Select Controller → M258 or LMC058 → IoDrvTM5SEAISG → StrainGaugeExt in the list, drag-and-drop the item onto the POU window.
2	Create the function block instance by clicking on: 
3	The inputs and outputs are detailed in the Description I/O Variables (<i>see page 17</i>).

StrainGaugeExt Function Block Presentation

Overview

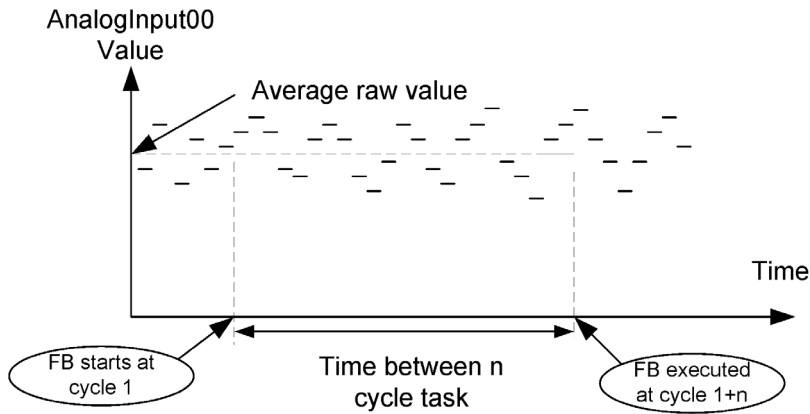
The **StrainGaugeExt** function block is an extended version of the **StrainGauge** function block, as it provides the capability to make continuous weight measurement on any type of bus (like TM5 and CANopen).

The **StrainGaugeExt** function block can be used with TM5SEAISG in local, remote and distributed architectures.

The **StrainGauge** function block has 3 functions:

- make an average measure of the TM5SEAISG input in a defined period
- define a linear calibration to match the needs of your process
- provide a calibrated measure

The average raw value is calculated with all the measures done by the TM5SEAISG module during a define number of task cycles. The number of task cycles is set with the `Cycle_Number` input of the function block.



Where n is the `Cycle_number` value.

StrainGaugeExt Function Block Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the Function and Function Block Representation (*see page 35*) chapter.

Description of I/O Variables

The table shows the input variables:

Input	Type	Initial	Comment
xEnable	BOOL	–	TRUE = action running. FALSE = action stopped, the outputs xDone, xBusy, xError and iError are reset.
AnalogInput	DINT	CST_ INVALID_ VALUE	Raw value given by StrainGauge module. To be mapped through a variable to AnalogInput00 in: <ul style="list-style-type: none"> • I/O Mapping of TM5SEAISG module or; • CANopen I/O Mapping of the TM5/TM7 DTM if StrainGauge module is used with TM5 CANopen Interface.
Tare_Enable	BOOL	FALSE	TRUE = enables the taring function.
Ref1_Enable	BOOL	FALSE	TRUE = enables the measure of the point reference number 1.
Ref2_Enable	BOOL	FALSE	TRUE = enables the measure of the point reference number 2.
Cycle_number	BYTE	1	Number of task cycles that is used to make an average measure of the raw value contained in AnalogInput00 (must be different than 0).
xContinuous	BOOL	FALSE	Running mode: <ul style="list-style-type: none"> • TRUE = Continuous measurement. • FALSE = Single measurement.
s_strainGaugeParameter	StrainGaugeParameter (<i>see page 33</i>)	–	Taring and calibration values.

The table shows the output variables:

Output	Type	Initial	Comment
xDone	BOOL	–	TRUE = indicates that the action is successfully completed. Function block execution is finished.
xBusy	BOOL	–	TRUE = indicates that the function block execution is in progress.
xError	BOOL	–	TRUE = indicates that an error was detected and the function block aborts the action. Function block execution is finished.
xReady	BOOL	FALSE	TRUE = indicates that the Calibrated_value is valid.
eError	StainGauge_Error <i>(see page 32)</i>	0	When xError is TRUE: type of the detected error.
Calibrated_value	DINT	CST_INVALID_VALUE	Value calculated after the calibration processing of the function block.

Section 2.2

Strain Gauge Legacy Function Block

What Is in This Section?


This section contains the following topics:

Topic	Page
Adding the StrainGauge Function Block	20
StrainGauge Function Block Presentation	21

Adding the StrainGauge Function Block

Procedure

Follow these steps to add and create the instance of a **StrainGauge** function block:

Step	Action
1	Select the Libraries tab in the Software Catalog and click Libraries . Select Controller → M258 or LMC058 → IoDrvTM5SEAISG → Legacy → StrainGauge in the list, drag-and-drop the item onto the POU window.
2	Create the function block instance by clicking on: 
3	The inputs and outputs are detailed in the Description I/O Variables (<i>see page 22</i>).

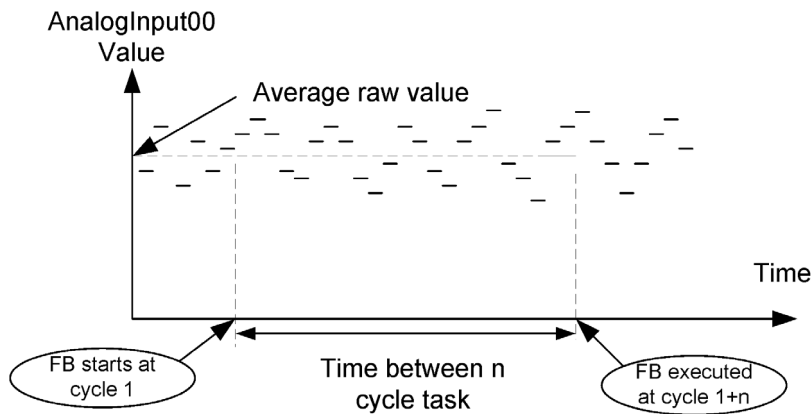
StrainGauge Function Block Presentation

Overview

The StrainGauge function block has 3 functions:

- make an average measure of the TM5SEAI5G input in a defined period
- define a linear calibration to match the needs of your process
- provide a calibrated mesure

The average raw value is calculated with all the mesures done by the TM5SEAI5G module during a define number of task cycles. The number of task cycles is set with the `Cycle_Number` input of the function block.



Where n is the `Cycle_number` value.

StrainGauge Function Block Representation



IL and ST Representation

To see the general representation in IL or ST language, refer to the Function and Function Block Representation ([see page 35](#)) chapter.

Description of I/O Variables

The following table describes the input variables:

Input	Type	Initial	Comment
xExecute	BOOL	–	On rising edge, starts the function block execution. On falling edge, resets the outputs of the function block when its execution terminates.
Module_Ref	TM5_STRAINGAUGE	–	Reference of the expansion electronic module TM5SEAISG.
Tare_Enable	BOOL	FALSE	TRUE = enables the taring function.
Ref1_Enable	BOOL	FALSE	TRUE = enables the measure of the point reference number 1.
Ref2_Enable	BOOL	FALSE	TRUE = enables the measure of the point reference number 2.
Cycle_number	DWORD	0	Number of task cycles that is used to make an average measure of the raw value contained in AnalogInput00 (must be different than 0).
s_strainGaugeParameter	StrainGaugeParameter <i>(see page 33)</i>	–	Taring and calibration values.

The following table describes the output variables:

Output	Type	Initial	Comment
xDone	BOOL	–	TRUE = indicates that the action is successfully completed. Function block execution is finished.
xBusy	BOOL	–	TRUE = indicates that the function block execution is in progress.
xError	BOOL	–	TRUE = indicates that an error was detected and the function block aborts the action. Function block execution is finished.
eError	StainGauge_Error <i>(see page 32)</i>	0	When xError is TRUE: type of the detected error.
Calibrated_value	DINT	FF80 0000 hex	Value calculated after the calibration processing of the function block <i>(see page 23)</i> .

Section 2.3

Calibrate Your System

What Is in This Section?

This section contains the following topics:

Topic	Page
Linear Calibration	24
Create the First Reference Point	25
Create the Second Reference Point	26
Taring Your System	27

Linear Calibration

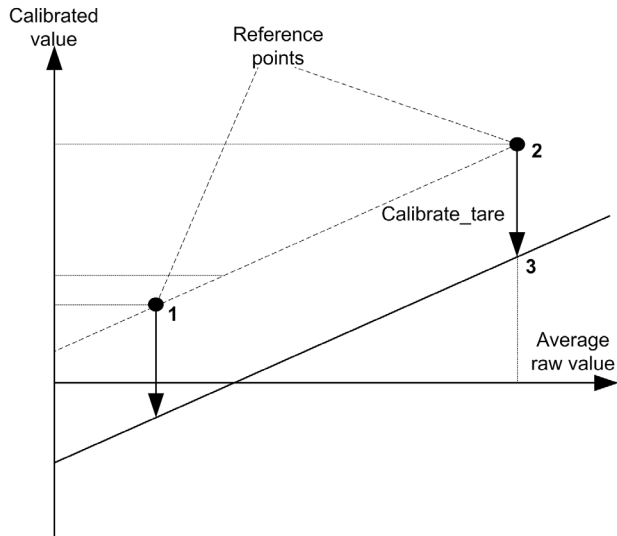
Overview

The TM5 `StrainGauge` function block provides a calibrated measurement. It is necessary to calibrate your system before starting any measurement.

The calibration of your system is done in 3 steps:

Step	Action
1	Define a first reference point.
2	Define a second reference point.
3	Define a tare.

The calibrated measure is done by linear interpolation:



The calibrated line is saved in a variable of type `StrainGaugeParameter` (*see page 33*).

NOTE: To define the calibration line, it is recommended to choose two reference points around the nominal measurement value. The first reference point at 10...20% of the nominal value and the second reference point at 50...60% of the nominal value.

Create the First Reference Point

Conditions

The following conditions must be respected:

- `Module_Ref` must have a correct value (*see page 22*)
- `Cycle_number` must have a correct value (*see page 22*)

Procedure

The procedure sets the `Raw_Ref1` field of the `s_strainGaugeParameter` structure.

Follow these steps to create the first reference point:

Step	Action
1	Create and stabilize the conditions that are representative of the measurement required for the first reference point.
2	Set the inputs of the <code>StrainGauge</code> function block to following values: Tare_Enable = 0 Ref1_Enable = 1 Ref2_Enable = 0
3	Set the function block input <code>xExecute</code> to 1.
4	When <code>xDone = 1</code> , <code>s_strainGaugeParameter.RawRef1</code> is set to the average value calculated by the function block.
5	Set the corresponding calibrated value you wish to associate with <code>Raw_Ref1</code> in the <code>s_strainGaugeParameter.Calibrate_Ref1</code> .

Create the Second Reference Point

Conditions

The following conditions must be respected:

- `Module_Ref` must have a correct value (see page 22)
- `Cycle_number` must have a correct value (see page 22)
- Reference 1 must be previously established

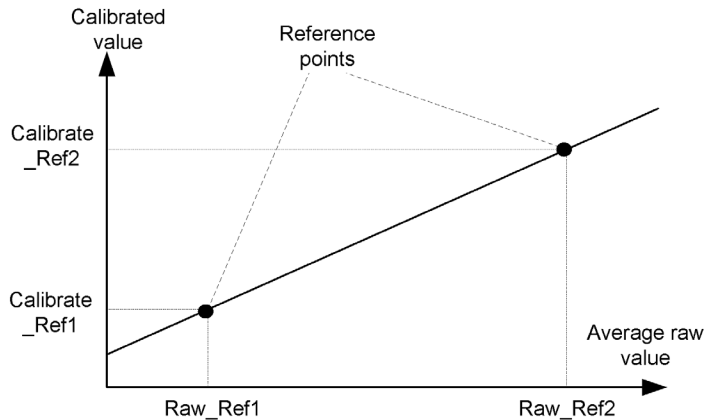
Procedure

The procedure sets the `Raw_Ref2` field of the `s_strainGaugeParameter` structure.

Follow these steps to create the first reference point:

Step	Action
1	Create and stabilize the conditions that are representative of the measurement required for the second reference point.
2	Set the inputs of the <code>StrainGauge</code> function block to following values: Tare_Enable = 0 Ref1_Enable = 0 Ref2_Enable = 1
3	Set the function block input <code>xExecute</code> to 1.
4	When <code>xDone = 1</code> , <code>s_strainGaugeParameter.RawRef2</code> is set to the average value calculated by the function block.
5	Set the corresponding calibrated value you wish to associate with <code>Raw_Ref2</code> in the <code>s_strainGaugeParameter.Calibrate_Ref2</code> .

Defining both reference points allows to establish the calibration line:



Taring Your System

Conditions

The following conditions must be respected:

- `Module_Ref` must have a correct value (*see page 22*)
- `Cycle_number` must have a correct value (*see page 22*)
- Reference 1 must be previously established
- Reference 2 must be previously established

Taring Procedure

This procedure allows you to create an offset to establish a net value when there is a load, or “tare”, measured and indicated by the TM5EASIG module.

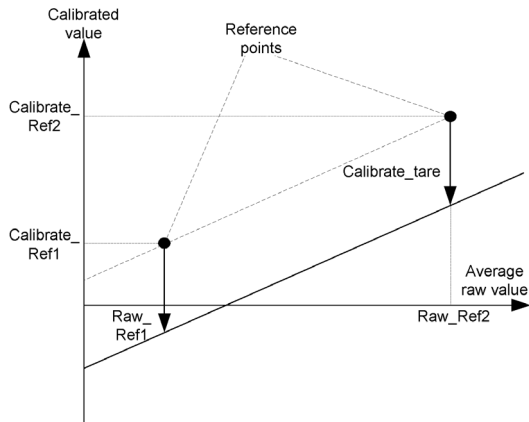
The procedure sets the `Calibrate_Tare` field of the `s_strainGaugeParameter` structure.

NOTE: The tare is derived from the calibrated line.

Follow these steps to tare a TM5EASIG electronic module:

Step	Action
1	Create and stabilize the conditions that are representative of the measurement required for the tare.
2	Set the inputs of the <code>StrainGauge</code> function block to following values: Tare_Enable = 1 Ref1_Enable = 0 Ref2_Enable = 0
3	Set the function block input <code>xExecute</code> to 1.
4	<code>s_strainGaugeParameter.Tare</code> is set to the calibrated value calculated by the function block.

An offset is created on the calibrated line previously defined by both reference points:



Section 2.4

Measure a Calibrated Value

Using Your System

Conditions

The following conditions must be respected:

- `Module_Ref` must have a correct value (*see page 22*)
- `Cycle_Number` must have a correct value (*see page 22*)
- Reference 1 must be previously established
- Reference 2 must be previously established

NOTE: If no calibration parameters or incorrect calibration parameters are provided, the function block returns an error (06 hex) Error Codes (*see page 32*).

Measuring Procedure

After you calibrated your system and set a tare value if necessary, this procedure is used to obtain the calibrated value as measured by the TM5SEASIG module and calculated by the function block.

Follow these steps to measure a value:

Step	Action
1	Set the function block with the following input value: Tare_Enable = 0 Ref1_Enable = 0 Ref2_Enable = 0 xContinuous = TRUE if you want a continuous measurement (extended FB only)
2	Set the function block input <code>xExecute</code> to 1.
3	When <code>xDone</code> = 1, the <code>Calibrated_value</code> output of the function block provides the calibrated value as measured by the TM5SEASIG module and calculated by the function block.

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	Data Types	31
B	Function and Function Block Representation	35

Appendix A

Data Types

Overview

This chapter describes the data types of the strain gauge library.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
StainGauge_Error: Error Codes	32
StrainGaugeParameter: Calibration Parameters	33

StainGauge_Error: Error Codes

Enumerated Type Description

The `StainGauge_Error` enumeration data type contains the following values:

Enumerator	Value	Description
CALIBRATION_OK	00 hex	Indicates that the measure is valid.
OVERFLOW_VALUE	01 hex	Indicates that the average raw value is above of the maximum value.
UNDERFLOW_VALUE	02 hex	Indicates that the average raw value is below of the minimum value.
INVALID_VALUE	03 hex	Indicates that the average raw value is invalid or the module is busy.
INVALID_CYCLETIME	04 hex	Indicates that the function block input <code>Cycle_number = 0</code> .
MULTIPLE_COMMAND	05 hex	Indicates that 2 of the function block inputs <code>Taring_enable</code> , <code>Ref1_enable</code> or <code>Ref2_enable</code> are set to 1.
INVALID_CALIBRATION_PARAMETERS	06 hex	Indicates that the calibration parameters (<i>see page 23</i>) are invalid.
INVALID_MODULE_REF	07 hex	Indicates that the the module reference on the <code>Module_Ref</code> input is incorrect.

StrainGaugeParameter: Calibration Parameters

Enumerated Type Description

The `StrainGaugeParameter` enumeration data type contains the following values:

Enumerator	Value	Description
<code>Calibrate_Tare</code>	DINT	Offset of the calibrated value.
<code>Calibrate_Ref1</code>	DINT	Calibrated value at reference point 1 of the calibration line.
<code>Calibrate_Ref2</code>	DINT	Calibrated value at reference point 2 of the calibration line.
<code>Raw_Ref1</code>	REAL	Average raw value at reference point 1 of the calibration line.
<code>Raw_Ref2</code>	REAL	Average raw value at reference point 2 of the calibration line.

Appendix B

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	36
How to Use a Function or a Function Block in IL Language	37
How to Use a Function or a Function Block in ST Language	41

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

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

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

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

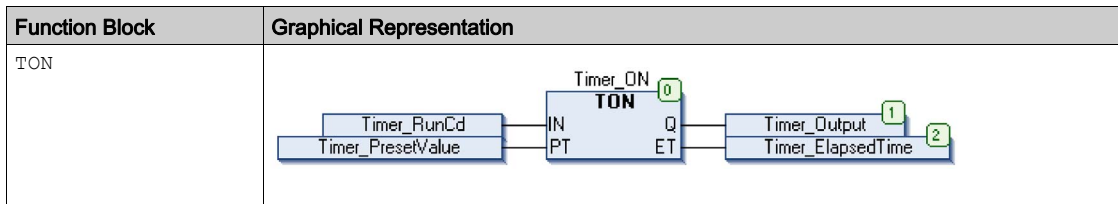
Function	Representation in POU IL Editor															
IL example of a function without input parameter: IsFirstMastCycle	<pre> 1 PROGRAM MyProgram_IL 2 VAR 3 FirstCycle: BOOL; 4 END_VAR 5 </pre> <hr/> <table border="1" data-bbox="391 467 974 576"> <tr> <td data-bbox="391 467 432 495">1</td> <td data-bbox="432 467 734 495">IsFirstMastCycle</td> <td data-bbox="734 467 974 495"></td> </tr> <tr> <td data-bbox="391 495 432 522"></td> <td data-bbox="432 495 734 522">ST</td> <td data-bbox="734 495 974 522">FirstCycle</td> </tr> </table>	1	IsFirstMastCycle			ST	FirstCycle									
1	IsFirstMastCycle															
	ST	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 9 </pre> <hr/> <table border="1" data-bbox="391 990 926 1169"> <tr> <td data-bbox="391 990 432 1018">1</td> <td data-bbox="432 990 679 1018">LD</td> <td data-bbox="679 990 926 1018">myDrift</td> </tr> <tr> <td data-bbox="391 1018 432 1045"></td> <td data-bbox="432 1018 679 1045">SetRTCDrift</td> <td data-bbox="679 1018 926 1045">myDay</td> </tr> <tr> <td data-bbox="391 1045 432 1073"></td> <td data-bbox="432 1045 679 1073"></td> <td data-bbox="679 1045 926 1073">myHour</td> </tr> <tr> <td data-bbox="391 1073 432 1101"></td> <td data-bbox="432 1073 679 1101"></td> <td data-bbox="679 1073 926 1101">myMinute</td> </tr> <tr> <td data-bbox="391 1101 432 1128"></td> <td data-bbox="432 1101 679 1128">ST</td> <td data-bbox="679 1101 926 1128">myDiag</td> </tr> </table>	1	LD	myDrift		SetRTCDrift	myDay			myHour			myMinute		ST	myDiag
1	LD	myDrift														
	SetRTCDrift	myDay														
		myHour														
		myMinute														
	ST	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. NOTE: The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POU's.
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a CAL instruction: <ul style="list-style-type: none"> ● Use the Input Assistant to select the FB (right-click and select Insert Box in the context menu). ● Automatically, the CAL instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: <ul style="list-style-type: none"> ● Values to inputs are set by " := ". ● Values to outputs are set by " => ".
4	In the CAL right-side field, replace ??? with the instance name.
5	Replace other ??? with an appropriate variable or immediate value.

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



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

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

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

General Information

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

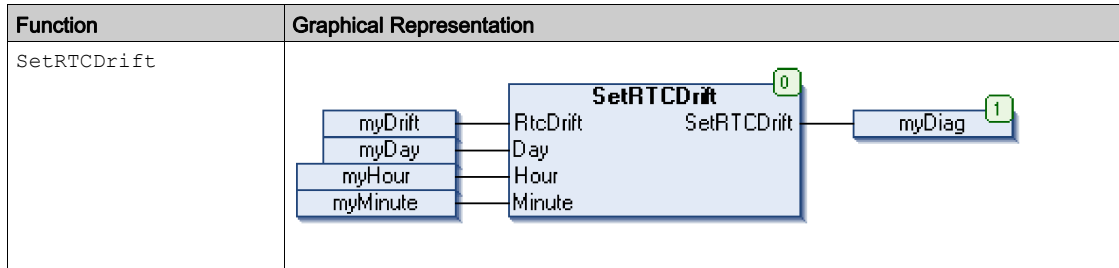
Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

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

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

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



The ST language of this function is the following:

Function	Representation in 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. NOTE: The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POU's.
2	Create the input and output variables and the instance required for the function block: <ul style="list-style-type: none"> • Input variables are the input parameters required by the function block • Output variables receive the value returned by the function block
3	Use the general syntax in the POU ST Editor for the ST language of a Function Block. The general syntax is: <pre>FunctionBlock_InstanceName (Input1:=VarInput1, Input2:=VarInput2, ... Ouput1=>VarOutput1, Ouput2=>VarOutput2, ...);</pre>

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

Function Block	Graphical Representation
TON	

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

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



B

byte

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

C

CFC

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

F

FB

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

function block diagram

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

I

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).

P

POU

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

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.

T

tare

Mass of the empty packaging of a product, derived from the gross weight to get the net weight.

task

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

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

A controller can have several tasks.

V

variable

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



C

calibrated measurements, Strain Gauge, *28*
calibration, overview of Strain Gauge, *24*
continuous measurement with StrainGauge-Ext function block, *16*

D

data types
 StrainGauge_Error, *32*
 StrainGaugeParameter, *33*

F

function blocks
 StrainGauge, *20, 21*
 StrainGaugeExt, *11, 15*
functions
 differences between a function and a function block, *36*
 how to use a function or a function block in IL language, *37*
 how to use a function or a function block in ST language, *41*

I

installing
 TM5SEAISG Strain Guide system, *11*
IoDrvTM5SEAISG, *11*

L

legacy Strain Gauge function block, *21*
library, IoDrvTM5SEAISG, *11*
linear calibration, Strain Gauge, *24*

M

measurement, continuous, *16*
measurements, Strain Gauge, *28*

O

one-shot measurement with StrainGauge function block, *21*

R

reference points, creating Strain Gauge, *25, 26*

S

single measurement with StrainGauge function block, *21*
Strain Gauge
 creating first reference point, *25*
 creating second reference point, *26*
 linear calibration of, *24*
 measuring with, *28*
 taring, *27*
StrainGauge function block
 adding, *20*
 inputs and outputs, *21*
StrainGauge_Error
 data types, *32*
StrainGaugeExt function block, *11*
 adding, *15*
 continuous measurements with, *16*
 inputs and outputs, *16*
StrainGaugeParameter
 calibration parameters, *24*
 data types, *33*

T

taring Strain Gauge system, *27*

task cycles, setting number of, *11*

TM5SEASIG

 creating measurement system with, *11*

 taring, *27*