Modicon TM5
Strain Gauge IoDrvTM5SEAISG Library Guide

05/2019

EIO0000003185.00

www.schneider-electric.com

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

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

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

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

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

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

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

About the Book

At a Glance

Document Scope

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

This documentation describes the function block and variables of the IoDrvTM5SEAISG strain gauge library.

In order to use this manual, you must:

- Have a thorough understanding of the TM5SEAISG, 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 V1.1.

Related Documents

Title of Documentation	Reference Number
Modicon M258 Logic Controller Programming Guide	*EIO0000000402 (Eng)*; *EIO0000000403 (Fre)*; *EIO0000000404 (Ger)*; *EIO0000000405 (Spa)*; *EIO0000000406 (Ita)*; *EIO0000000407 (Chs)*
Modicon LMC058 Motion Controller Programming Guide	*EIO0000000408 (Eng)*; *EIO0000000409 (Fre)*; *EIO0000000410 (Ger)*; *EIO0000000411 (Spa)*; *EIO0000000412 (Ita)*; *EIO0000000413 (Chs)*
Modicon TM5 Expansion Modules Configuration Programming Guide	*EIO0000000420 (Eng)*, *EIO0000000421 (Fre)*, *EIO0000000422 (Ger)*, *EIO0000000423 (Spa)*, *EIO0000000424 (Ita)*, *EIO0000000425 (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.schneider-electric.com/en/download

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

Chapter 1
General Overview

Create your Measurement System

Overview

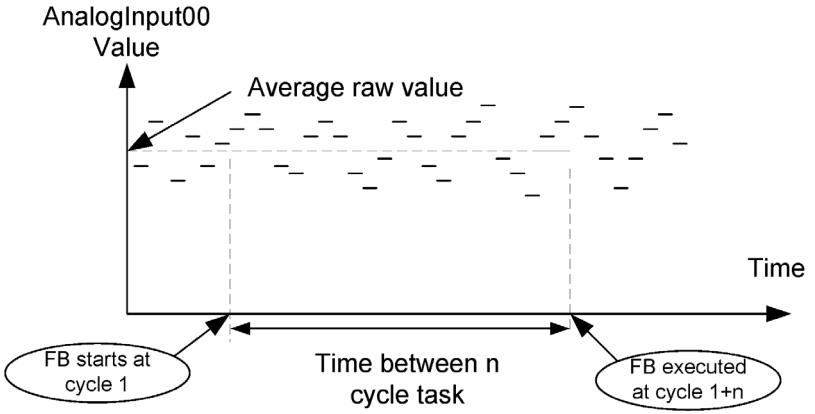
The full-bridge strain gauge sensor provides an electrical signal that the TM5SEAISG module converts to a raw numerical value. The raw value is then processed with the use of the StrainGaugeExt function block *(see page 16)* which provides a calibrated value.

This 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 mesure

NOTE: The Strain Gauge function block is not automatically declared when you add TM5SEAISG to the controller.

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.

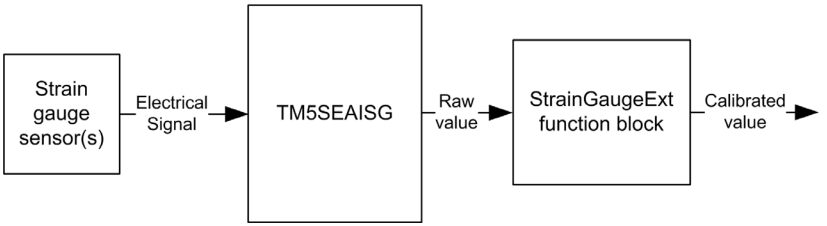


Where n is the **Cycle_number** value.

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 *(see Modicon TM5, Analog I/O Modules, Hardware Guide)*.

Measurement Cycle

The measurement system is shown in the following diagram:



Your measurement system is created through the configuration *(see Modicon TM5, Expansion Modules Configuration, Programming Guide)* 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 2.1
Strain Gauge Function Block

What Is in This Section?

This section contains the following topics:

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 *(see page 17)*.

StrainGaugeExt Function Block Presentation

Overview

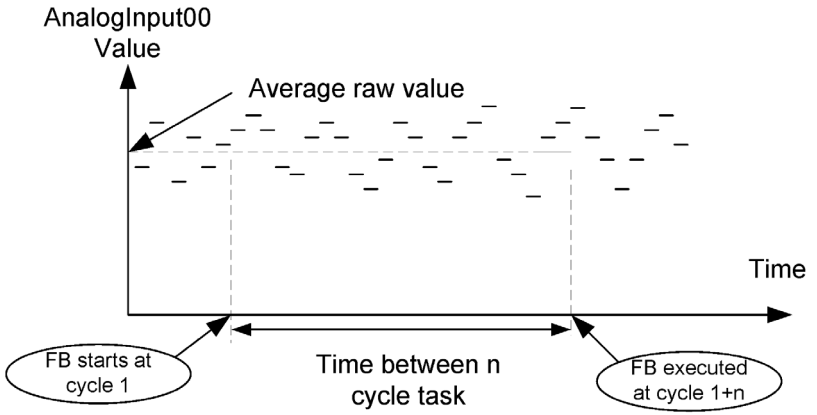
The **StrainGaugeExt** function block is an extended version of the **StrainGauge** function block, as it provides the capability to make continuous weight measurement on any type of bus (like TM5 and CANopen).

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 37)* 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: • `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: • `TRUE` = Continuous measurement. • `FALSE` = Single measurement.
`s_strainGaugeParameter`	`StrainGaugeParameter` *(see page 35)*	–	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 *(see page 34)*	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:

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 *(see page 22)*.

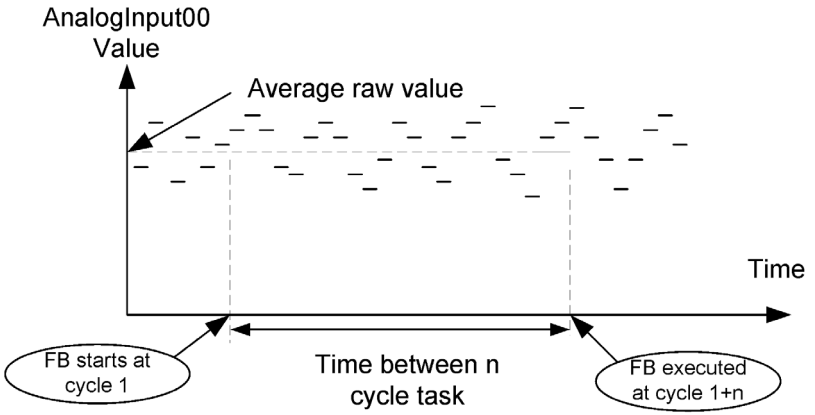
StrainGauge Function Block Presentation

Overview

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 mesure

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.

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 37)* 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 *(see page 35)*	–	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 *(see page 34)*	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 *(see page 23)*.

Section 2.3
Calibrate Your System

What Is in This Section?

This section contains the following topics:

Linear Calibration

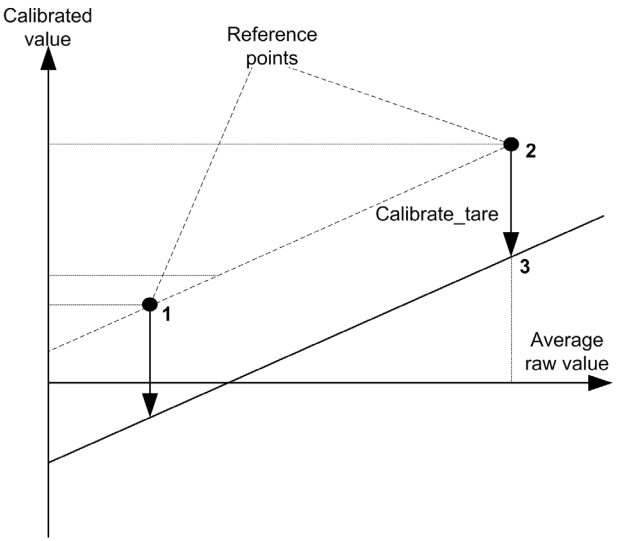
Overview

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 35)*.

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 `StrainGauge` function block to following values: **Tare_Enable** = 0 **Ref1_Enable** = 1 **Ref2_Enable** = 0
3	Set the function block input `xExecute` to 1.
4	When `xDone` = 1, `s_strainGaugeParameter.RawRef1` is set to the average value calculated by the function block.
5	Set the corresponding calibrated value you wish to associate with `Raw_Ref1` in the `s_strainGaugeParameter.Calibrate_Ref1`.

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

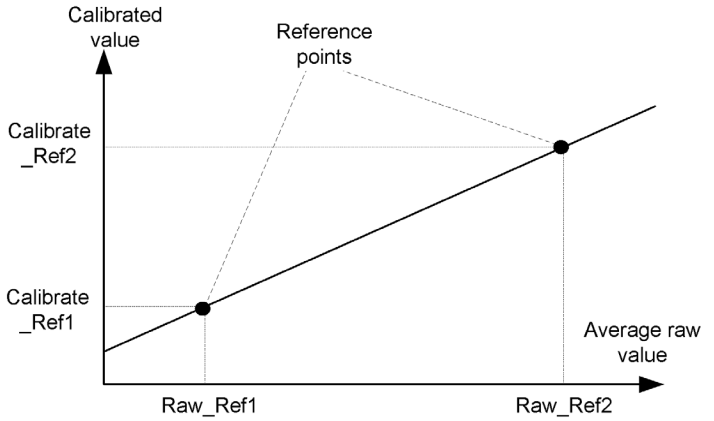
Procedure

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 `StrainGauge` function block to following values: **Tare_Enable** = 0 **Ref1_Enable** = 0 **Ref2_Enable** = 1
3	Set the function block input `xExecute` to 1.
4	When `xDone` = 1, `s_strainGaugeParameter.RawRef2` is set to the average value calculated by the function block.
5	Set the corresponding calibrated value you wish to associate with `Raw_Ref2` in the `s_strainGaugeParameter.Calibrate_Ref2`.

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 TMSEAISG module.

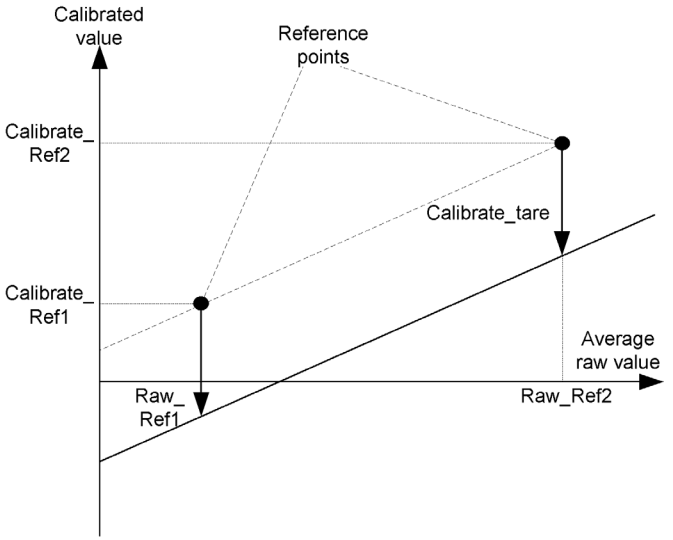
The procedure sets the `Calibrate_Tare` field of the `s_strainGaugeParameter` structure.

NOTE: The tare is derived from the calibrated line.

Follow these steps to tare a TM5SEAISG 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 `StrainGauge` function block to following values: **Tare_Enable** = 1 **Ref1_Enable** = 0 **Ref2_Enable** = 0
3	Set the function block input `xExecute` to 1.
4	`s_strainGaugeParameter.Tare` 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 34)*.

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 TM5SEAISG 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 `xExecute` to 1.
3	When `xDone` = 1, the `Calibrated_value` output of the function block provides the calibrated value as measured by the TM5SEAISG 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:

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:

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 Cycle_number = 0.
MULTIPLE_COMMAND	05 hex	Indicates that 2 of the function block inputs Taring_enable, Ref1_enable or Ref2_enable are set to 1.
INVALID_CALIBRATION_PARAMETERS	06 hex	Indicates that the calibration parameters *(see page 23)* are invalid.
INVALID_MODULE_REF	07 hex	Indicates that the the module reference on the Module_Ref input is incorrect.

`StrainGaugeParameter`: Calibration Parameters

Enumerated Type Description

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

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

Differences Between a Function and a Function Block

Function

A function:

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

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

Function Block

A function block:

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

Examples: timers, counters

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

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

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

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

General Information

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

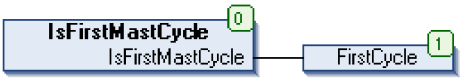
Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

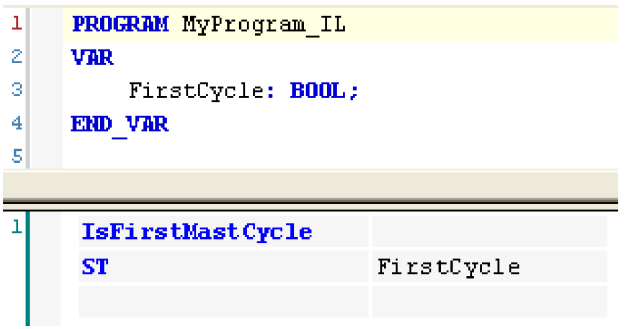
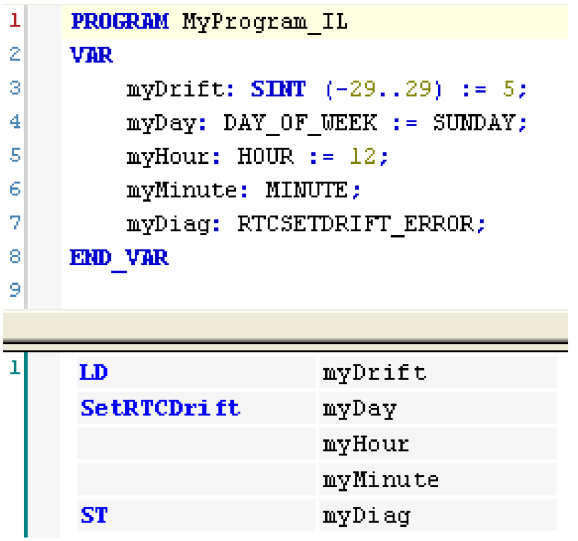
This procedure describes how to insert a function in IL language:

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

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

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

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

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

Using a Function Block in IL Language

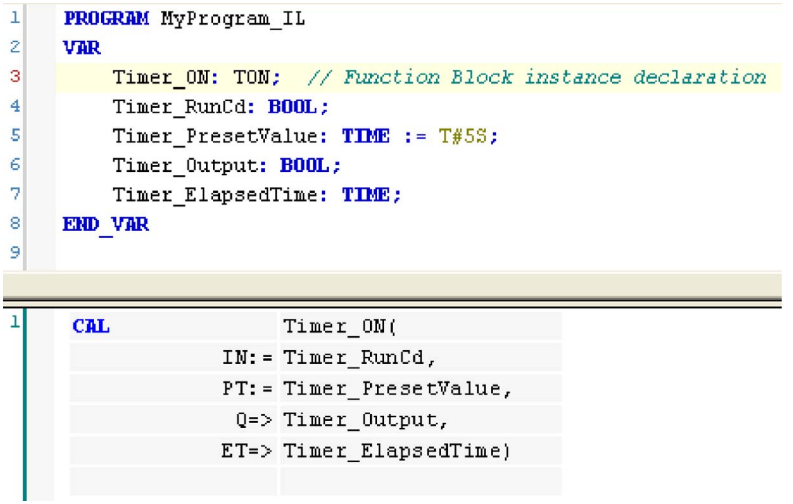
This procedure describes how to insert a function block in IL language:

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

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

Function Block	Graphical Representation
TON	

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

Function Block	Representation in POU IL Editor
TON	

```
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
```

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

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

General Information

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

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

Using a Function in ST Language

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

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

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

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

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

Using a Function Block in ST Language

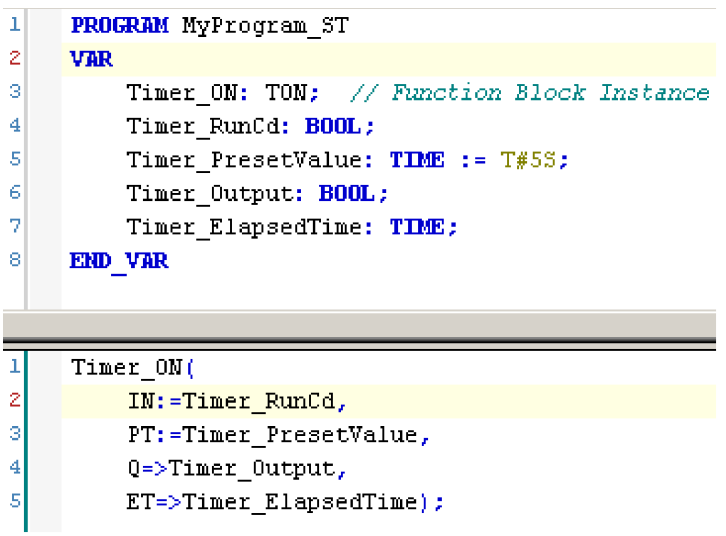
This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation *(see EcoStruxure Machine Expert, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block
3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is: `FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,... Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

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

Function Block	Graphical Representation
`TON`	

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

Function Block	Representation in POU ST Editor
TON	

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

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

Glossary

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. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs 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.

Index