

# VIPA SPEED7 Library

OPL\_SP7-LIB | SW90LS0MA V10.005 | Manual

HB00 | OPL\_SP7-LIB | SW90LS0MA V10.005 | en | 18-27

Block library - Device Specific



VIPA GmbH  
Ohmstr. 4  
91074 Herzogenaurach  
Telephone: +49 9132 744-0  
Fax: +49 9132 744-1864  
Email: [info@vipa.com](mailto:info@vipa.com)  
Internet: [www.vipa.com](http://www.vipa.com)

## Table of contents

<b>1</b>	<b>General</b> .....	<b>4</b>
1.1	Copyright © VIPA GmbH .....	4
1.2	About this manual.....	5
<b>2</b>	<b>Important notes</b> .....	<b>6</b>
2.1	General.....	6
2.2	Internally used blocks.....	6
2.3	No optimized block access.....	7
2.4	Declaration types.....	7
<b>3</b>	<b>Include library</b> .....	<b>8</b>
3.1	Integration into Siemens SIMATIC Manager.....	8
3.2	Integration into Siemens TIA Portal.....	9
<b>4</b>	<b>Block parameters</b> .....	<b>10</b>
4.1	HW identifier - HW_ID.....	10
4.2	General and Specific Error Information RET_VAL.....	10
<b>5</b>	<b>Device Specific</b> .....	<b>13</b>
5.1	Frequency Measurement.....	13
5.1.1	FC 300 ... 303 - Frequency measurement SLIO consistent.....	13
5.1.2	FC 300 - FM_SET_CONTROL - Control frequency measurement consistent.....	13
5.1.3	FC 301 - FM_GET_PERIOD - Calculate period duration consistent.....	15
5.1.4	FC 302 - FM_GET_FREQUENCY - Calculate frequency consistent.....	17
5.1.5	FC 303 - FM_GET_SPEED - Calculate rotational speed consistent.....	20
5.1.6	FC 310 ... 313 - Frequency measurement SLIO.....	22
5.1.7	FC 310 - FM_CONTROL - Control frequency measurement.....	23
5.1.8	FC 311 - FM_CALC_PERIOD - Calculate period duration .....	24
5.1.9	FC 312 - FM_CALC_FREQUENCY - Calculate frequency.....	26
5.1.10	FC 313 - FM_CALC_SPEED - Calculate rotational speed.....	28
5.2	Energy Measurement.....	30
5.2.1	Overview.....	30
5.2.2	FB 325 - EM_COM_R1 - Communication with 031-1PAxx.....	34
5.2.3	UDT 325 - EM_DATA_R1 - Data structure for FB 325.....	34
5.3	Motion Modules.....	38
5.3.1	Overview.....	38
5.3.2	FB 320 - ACYC_RW - Acyclic access to the System SLIO motion module.....	40
5.3.3	FB 321 - ACYC_DS - Acyclic parametrization System SLIO motion module.....	43
5.3.4	UDT 321 - ACYC_OBJECT-DATA - Data structure for FB 321.....	46
5.4	RAM to WLD - "WLD".....	47
5.4.1	FB 240 - RAM_to_s7prog.wld - RAM to s7prog.wld.....	47
5.4.2	FB 241 - RAM_to_autoload.wld - RAM to autoload.wld.....	47
5.5	Onboard I/O System 100V.....	48
5.5.1	SFC 223 - PWM - Pulse duration modulation.....	48
5.5.2	SFC 224 - HSC - High-speed-Counter.....	50
5.5.3	SFC 225 - HF_PWM - HF pulse duration modulation.....	52

# 1 General

## 1.1 Copyright © VIPA GmbH

### All Rights Reserved

This document contains proprietary information of VIPA and is not to be disclosed or used except in accordance with applicable agreements.

This material is protected by the copyright laws. It may not be reproduced, distributed, or altered in any fashion by any entity (either internal or external to VIPA), except in accordance with applicable agreements, contracts or licensing, without the express written consent of VIPA and the business management owner of the material.

For permission to reproduce or distribute, please contact: VIPA, Gesellschaft für Visualisierung und Prozessautomatisierung mbH Ohmstraße 4, D-91074 Herzogenaurach, Germany

Tel.: +49 9132 744 -0

Fax.: +49 9132 744-1864

E-Mail: [info@vipa.de](mailto:info@vipa.de)

<http://www.vipa.com>



*Every effort has been made to ensure that the information contained in this document was complete and accurate at the time of publishing. Nevertheless, the authors retain the right to modify the information.*

*This customer document describes all the hardware units and functions known at the present time. Descriptions may be included for units which are not present at the customer site. The exact scope of delivery is described in the respective purchase contract.*

### CE Conformity Declaration

Hereby, VIPA GmbH declares that the products and systems are in compliance with the essential requirements and other relevant provisions. Conformity is indicated by the CE marking affixed to the product.

### Conformity Information

For more information regarding CE marking and Declaration of Conformity (DoC), please contact your local VIPA customer service organization.

### Trademarks

VIPA, SLIO, System 100V, System 200V, System 300V, System 300S, System 400V, System 500S and Commander Compact are registered trademarks of VIPA Gesellschaft für Visualisierung und Prozessautomatisierung mbH.

SPEED7 is a registered trademark of profichip GmbH.

SIMATIC, STEP, SINEC, TIA Portal, S7-300, S7-400 and S7-1500 are registered trademarks of Siemens AG.

Microsoft and Windows are registered trademarks of Microsoft Inc., USA.

Portable Document Format (PDF) and Postscript are registered trademarks of Adobe Systems, Inc.

All other trademarks, logos and service or product marks specified herein are owned by their respective companies.

**Information product support**

Contact your local VIPA Customer Service Organization representative if you wish to report errors or questions regarding the contents of this document. If you are unable to locate a customer service centre, contact VIPA as follows:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Telefax: +49 9132 744-1204

E-Mail: [documentation@vipa.de](mailto:documentation@vipa.de)

**Technical support**

Contact your local VIPA Customer Service Organization representative if you encounter problems with the product or have questions regarding the product. If you are unable to locate a customer service centre, contact VIPA as follows:

VIPA GmbH, Ohmstraße 4, 91074 Herzogenaurach, Germany

Tel.: +49 9132 744-1150 (Hotline)

E-Mail: [support@vipa.de](mailto:support@vipa.de)

## 1.2 About this manual

**Objective and contents**

The manual describes the block library 'Device Specific' from VIPA:

- It contains a description of the structure, project implementation and usage in several programming systems.
- The manual is targeted at users who have a background in automation technology.
- The manual is available in electronic form as PDF file. This requires Adobe Acrobat Reader.
- The manual consists of chapters. Every chapter provides a self-contained description of a specific topic.
- The following guides are available in the manual:
  - An overall table of contents at the beginning of the manual
  - References with pages numbers

**Icons Headings**

Important passages in the text are highlighted by following icons and headings:

**DANGER!**

Immediate or likely danger. Personal injury is possible.

**CAUTION!**

Damages to property is likely if these warnings are not heeded.



*Supplementary information and useful tips.*

Internally used blocks

## 2 Important notes

### 2.1 General



*In the following, you will find important notes, which must always be observed when using the blocks.*

### 2.2 Internally used blocks



#### CAUTION!

The following blocks are used internally and must not be overwritten! The direct call of an internal block leads to errors in the corresponding instance DB! Please always use the corresponding function for the call.

FC/SFC	Designation	Description
FC/SFC 192	CP_S_R	is used internally for FB 7 and FB 8
FC/SFC 196	AG_CNTRL	is used internally for FC 10
FC/SFC 200	AG_GET	is used internally for FB/SFB 14
FC/SFC 201	AG_PUT	is used internally for FB/SFB 15
FC/SFC 202	AG_BSEND	is used internally for FB/SFB 12
FC/SFC 203	AG_BRCV	is used internally for FB/SFB 13
FC/SFC 204	IP_CONF	is used internally for FB 55 IP_CONF
FC/SFC 205	AG_SEND	is used internally for FC 5 AG_SEND
FC/SFC 206	AG_RECV	is used internally for FC 6 AG_RECV
FC/SFC 253	IBS_ACCESS	is used internally for SPEED bus INTERBUS masters
SFB 238	EC_RWOD	is used internally for EtherCAT Communication
SFB 239	FUNC	is used internally for FB 240, FB 241

## 2.3 No optimized block access



Please note that the blocks of this library do not support optimized block access! When using instance blocks and data blocks, the optimized block access must be deactivated!

### Set block access

1. Open in the Siemens TIA Portal *Project navigation* the 'Program blocks'.
2. Select the block for which you want to change the block access and select 'Context menu → Properties'.  
⇒ The "Properties" dialog of the block opens.
3. Select 'Attributes'.
4. Deactivate the parameter 'Optimized block access'.
5. Confirm with [OK].

More information can be found in the manual of the Siemens TIA Portal.

## 2.4 Declaration types

Please note that the spellings of the declaration types in Siemens STEP7 and TIA Portal differ. This documentation uses the notation for Siemens STEP7. A comparison of the spellings can be found in the following table.

Siemens TIA Portal	Siemens STEP7
Input	IN
Output	OUT
InOut	IN_OUT
Static	STAT
Temp	TEMP

### 3 Include library

#### Block library 'Device Specific'

The block library can be found for download in the 'Service/Support' area of [www.vipa.com](http://www.vipa.com) at 'Downloads → VIPA Lib' as 'Block library Device Specific - SW90LS0MA'. The library is available as packed zip file. As soon as you want to use these blocks you have to import them into your project.

The following block libraries are available

File	Description
DeviceSpecific_S7_V0005.zip	<ul style="list-style-type: none"> <li>■ Block library for Siemens SIMATIC Manager.</li> <li>■ For use in CPUs from VIPA or S7-300 CPUs from Siemens.</li> </ul>
DeviceSpecific_TIA_V0006.zip	<ul style="list-style-type: none"> <li>■ Block library for Siemens TIA Portal V14.</li> <li>■ For use in CPUs from VIPA or S7-300 CPUs from Siemens.</li> </ul>
DeviceSpecific_TIA_1500_V0002.zip	<ul style="list-style-type: none"> <li>■ Block library for Siemens TIA Portal V14.</li> <li>■ For use in S7-1500 CPUs from Siemens.</li> </ul>

#### 3.1 Integration into Siemens SIMATIC Manager

##### Overview

The integration into the Siemens SIMATIC Manager requires the following steps:

1. ➔ Load ZIP file
2. ➔ "Retrieve" the library
3. ➔ Open library and transfer blocks into the project

##### Load ZIP file

- ➔ Navigate on the web page to the desired ZIP file, load and store it in your work directory.

##### Retrieve library

1. ➔ Start the Siemens SIMATIC Manager with your project.
2. ➔ Open the dialog window for ZIP file selection via 'File → Retrieve'.
3. ➔ Select the according ZIP file and click at [Open].
4. ➔ Select a destination folder where the blocks are to be stored.
5. ➔ Start the extraction with [OK].

##### Open library and transfer blocks into the project

1. ➔ Open the library after the extraction.
2. ➔ Open your project and copy the necessary blocks from the library into the directory "blocks" of your project.
  - ⇒ Now you have access to the VIPA specific blocks via your user application.



Are FCs used instead of SFCs, so they are supported by the VIPA CPUs starting from firmware 3.6.0.



## 3.2 Integration into Siemens TIA Portal

### Overview

The integration into the Siemens TIA Portal requires the following steps:

1. ➤ Load ZIP file
2. ➤ Unzip the Zip file
3. ➤ "Retrieve" the library
4. ➤ Open library and transfer blocks into the project

### Load ZIP file

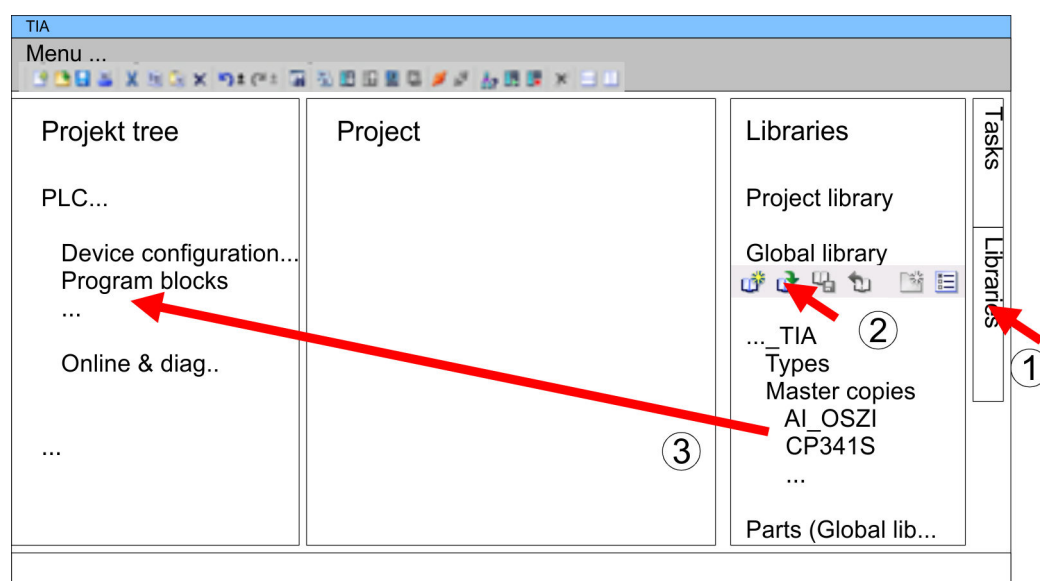
1. ➤ Navigate on the web page to the ZIP file, that matches your version of the program.
2. ➤ Load and store it in your work directory.

### Unzip the Zip file

- Unzip the zip file to a work directory of the Siemens TIA Portal with your unzip application.

### Open library and transfer blocks into the project

1. ➤ Start the Siemens TIA Portal with your project.
2. ➤ Switch to the *Project view*.
3. ➤ Choose "Libraries" from the task cards on the right side.
4. ➤ Click at "Global libraries".
5. ➤ Click at "Open global libraries".
6. ➤ Navigate to your work directory and load the file ...\_TIA.al1x.



7. ➤ Copy the necessary blocks from the library into the "Program blocks" of the *Project tree* of your project. Now you have access to the VIPA specific blocks via your user application.

## 4 Block parameters

### 4.1 HW identifier - HW\_ID

#### HW identifier

- The parameter *HW\_ID* to preset the *HW identifier* is only available in S7-1500 CPUs from Siemens.
- When configuring a hardware component, a hardware identifier is automatically assigned as *HW identifier* for each object of the hardware configuration.
- The *HW identifier* is for modules, ports, interfaces and I/O areas of bus systems.
- The *HW identifier* is a decimal integer constant of data type HW\_IO.
- The *HW identifier* does not distinguish between input and output ranges.
- You can use the *HW identifier* to address the corresponding hardware components.

#### Determine *HW identifier*

You can determine the *HW identifier* for the respective component using the following procedure:

1. ➤ Open in the *Project tree* the '*Device configuration*'.
2. ➤ Select the desired hardware component whose *HW identifier* you want to determine.
3. ➤ Click in the *Inspector* window at '*General*'.
  - ⇒ The '*HW identifier*' is shown. These can be used for the parameter *HW\_ID* when the blocks are connected.

#### *HW identifier* and system constants

You can also determine the *HW identifier* using the '*System constants*'. Via the '*System constants*' in the *Inspector* window all the HW identifiers of an object, which is selected in the device view, are listed with *Name* and *Type*. *Name* and *Type* are automatically generated when assigning the HW identifier. Here *Name* has a hierarchical structure with a maximum of 4 hierarchical levels, with each level separated by a "~". The name of the component of the corresponding hierarchy level can be changed at any time via the properties.

#### *HW identifier* in the user program

- When creating your user program, you can assign the corresponding hardware component from a list of all possible hardware components by double-clicking on the corresponding input or output parameter.
- In the case of a hardware interrupt, you can use the start information to determine the *HW identifier* as the '*ID*' of the hardware component that triggers the interrupt.

### 4.2 General and Specific Error Information RET\_VAL

#### Overview

The return value *RET\_VAL* of a system function provides one of the following types of error codes:

- A *general error code*, that relates to errors that can occur in anyone SFC.
- A *specific error code*, that relates only to the particular SFC.

Although the data type of the output parameter *RET\_VAL* is integer (INT), the error codes for system functions are grouped according to hexadecimal values.

If you want to examine a return value and compare the value with the error codes, then display the error code in hexadecimal format.

**RET\_VAL (Return value)** The table below shows the structure of a system function error code:

Bit	Description
7 ... 0	Event number or error class and single error
14 ... 8	Bit 14 ... 8 = "0": <b>Specific error code</b> The specific error codes are listed in the descriptions of the individual SFCs. Bit 14 ... 8 > "0": <b>General error code</b> The possible general error codes are shown
15	Bit 15 = "1": indicates that an error has occurred.

### Specific error code

This error code indicates that an error pertaining to a particular system function occurred during execution of the function.

A specific error code consists of the following two numbers:

- Error class between 0 and 7
- Error number between 0 and 15

Bit	Description
3 ... 0	Error number
6 ... 4	Error class
7	Bit 7 = "1"
14 ... 8	Bit 14 ... 8 = "0"
15	Bit 15 = "1": indicates that an error has occurred.

### General error codes RET\_VAL

The parameter *RET\_VAL* of some SFCs only returns general error information. No specific error information is available.

The general error code contains error information that can result from any system function. The general error code consists of the following two numbers:

- A parameter number between 1 and 111, where 1 indicates the first parameter of the SFC that was called, 2 the second etc.
- An event number between 0 and 127. The event number indicates that a synchronous fault has occurred.

Bit	Description
7 ... 0	Event number
14 ... 8	Parameter number
15	Bit 15 = "1": indicates that an error has occurred.

### General error codes

The following table explains the general error codes associated with a return value. Error codes are shown as hexadecimal numbers. The x in the code number is only used as a placeholder. The number represents the parameter of the system function that has caused the error.

Error code	Description
8x7Fh	Internal Error. This error code indicates an internal error at parameter x. This error did not result from the actions if the user and he/she can therefore not resolve the error.
8x01h	Illegal syntax detection for an ANY parameter.
8x22h	Area size error when a parameter is being read.
8x23h	Area size error when a parameter is being written. This error code indicates that parameter x is located either partially or fully outside of the operand area or that the length of the bit-field for an ANY-parameter is not divisible by 8.
8x24h	Area size error when a parameter is being read.
8x25h	Area size error when a parameter is being written. This error code indicates that parameter x is located in an area that is illegal for the system function. The description of the respective function specifies the areas that are not permitted for the function.
8x26h	The parameter contains a number that is too high for a time cell. This error code indicates that the time cell specified in parameter x does not exist.
8x27h	The parameter contains a number that is too high for a counter cell (numeric fields of the counter). This error code indicates that the counter cell specified in parameter x does not exist.
8x28h	Orientation error when reading a parameter.
8x29h	Orientation error when writing a parameter. This error code indicates that the reference to parameter x consists of an operand with a bit address that is not equal to 0.
8x30h	The parameter is located in the write-protected global-DB.
8x31h	The parameter is located in the write-protected instance-DB. This error code indicates that parameter x is located in a write-protected data block. If the data block was opened by the system function itself, then the system function will always return a value 8x30h.
8x32h	The parameter contains a DB-number that is too high (number error of the DB).
8x34h	The parameter contains a FC-number that is too high (number error of the FC).
8x35h	The parameter contains a FB-number that is too high (number error of the FB). This error code indicates that parameter x contains a block number that exceeds the maximum number permitted for block numbers.
8x3Ah	The parameter contains the number of a DB that was not loaded.
8x3Ch	The parameter contains the number of a FC that was not loaded.
8x3Eh	The parameter contains the number of a FB that was not loaded.
8x42h	An access error occurred while the system was busy reading a parameter from the peripheral area of the inputs.
8x43h	An access error occurred while the system was busy writing a parameter into den peripheral area of the outputs.
8x44h	Error during the n-th ( $n > 1$ ) read access after an error has occurred.
8x45h	Error during the n-th ( $n > 1$ ) write access after an error has occurred. This error code indicates that access was denied to the requested parameter.

## 5 Device Specific

### 5.1 Frequency Measurement

#### 5.1.1 FC 300 ... 303 - Frequency measurement SLIO consistent

##### Overview

The following VIPA specific functions are used to control the System SLIO frequency measurement modules, which are connected via PROFIBUS, PROFINET or EtherCAT. The usage with EtherCAT is only possible at an EtherCAT CPU from VIPA. By this functions SFC 14 - DPRD\_DAT respectively SFC 15 - DPWR\_DAT for consistent read respectively write access to the data are internally called. Error messages of these blocks are reported by the parameter *ERROR*.

Function	Symbol	Comment
FC 300	FM_SET_CONTROL	Function to control the frequency measurement with integrated consistent access.
FC 301	FM_GET_PERIOD	Function to calculate the period duration with integrated consistent access.
FC 302	FM_GET_FREQUENCY	Function to calculate the frequency with integrated consistent access.
FC 303	FM_GET_SPEED	Function to calculate the rotational speed with integrated consistent access.

#### 5.1.2 FC 300 - FM\_SET\_CONTROL - Control frequency measurement consistent

##### Description

The System SLIO Frequency measurement module is controlled by the FC 300 FM\_SET\_CONTROL. By this function the SFC 15 - DPWR\_DAT for consistent write access of data is called. Here error messages of the block are reported by *ERROR*.

##### Parameters

Parameter	Declaration	Data type	Memory block	Description
ENABLE_FM	INPUT	BOOL	I, Q, M, D, L	Enable frequency measurement
LADDR_OUT / HW_ID	INPUT	WORD / HW_IO	I, Q, M, D, L	<ul style="list-style-type: none"> <li>■ LADDR_OUT           <ul style="list-style-type: none"> <li>– Logical base output address of the frequency measurement module.</li> <li>– When used in CPUs from VIPA or in S7-300 CPUs from Siemens.</li> </ul> </li> <li>■ HW_ID           <ul style="list-style-type: none"> <li>– <i>HW identifier</i> to address the frequency measurement module.</li> <li>– When used in S7-1500 CPUs from Siemens.</li> </ul> </li> </ul>
PRESET_CH0	INPUT	DINT	I, Q, M, D, L	Channel 0: Measurement period
PRESET_CH1	INPUT	DINT	I, Q, M, D, L	Channel 1: Measurement period
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)

**ENABLE\_FM**

With setting *ENABLE\_FM* the *measuring periods*, which were preset by PRESET\_CH0/1, are transferred to the channels and the measurement of both channels are started. Both frequency meters are stopped by resetting *ENABLE\_FM*.



Only while *ENABLE\_FM* is set, evaluated values can be retrieved from the module. Otherwise you get the error message that the channels are disabled.

**LADDR\_OUT**

Peripheral address:

- This parameter is available in CPUs from VIPA or in S7-300 CPUs from Siemens.
- Configured base address of the output area of the System SLIO frequency measurement module, which is to be written to. The address is hexadecimal.
- (Example: Address 100: *LADDR\_OUT*: = W#16#64)

**HW\_ID**

HW identifier:

- This parameter is only available in S7-1500 CPUs of Siemens.
- Enter at *HW\_ID* the *HW identifier*, with which your module can be addressed accordingly. ↪ Chapter 4.1 'HW identifier - HW\_ID' on page 10

**PRESET\_CHx**

Enter here the measurement period in  $\mu\text{s}$  for the corresponding channel.

Range of values:  $1\mu\text{s} \dots 8\,388\,607\mu\text{s}$

**DONE**

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

**ERROR (Return value)**

The following code can be reported:

Code	Description
0x0000	No error
0x80D2	Channel 0: Input value measurement period $\leq 0$
0x80D3	Channel 1: Input value measurement period $\leq 0$
0x80D4	Channel 0: Input value measurement period $> 8\,388\,607\mu\text{s}$
0x80D5	Channel 1: Input value measurement period $> 8\,388\,607\mu\text{s}$

**Errors of the internally called SFC 15**

Code	Description
0x808x	System error on the bus coupler
0x8090	<i>LADDR_OUT</i> is wrong, possible reasons: <ul style="list-style-type: none"> <li>■ there is no module configured on this address</li> <li>■ limitation of the length of consistent data was not considered</li> <li>■ Basic address in parameter <i>LADDR_OUT</i> was not entered in hexadecimal type</li> </ul>
0x8093	There is no bus coupler existing for <i>LADDR_OUT</i> , from which consistent data can be read.
0x80A0	An access error was detected during peripheral access.
0x80B0	System error on the bus coupler
0x80B1	Specified length of the source area does not correspond to the configured user data length.
0x80B2	System error on the bus coupler
0x80B3	System error on the bus coupler
0x80C1	The data from the previous read request on the module are not processed by the module, yet.
0x80C2	System error on the bus coupler
0x80Fx	System error on the bus coupler
0x85xy	System error on the bus coupler
0x8xyy	General error information <a href="#">↪ Chapter 4.2 'General and Specific Error Information RET_VAL' on page 10</a>

**5.1.3 FC 301 - FM\_GET\_PERIOD - Calculate period duration consistent****Description**

With the FC 301 FM\_GET\_PERIOD, you can calculate the period duration of the input signals of both channels of the System SLIO frequency measurement module. By this function internally SFC 14 - DPRD\_DAT for consistent reading of user data is called. Here, the error messages of the function block are returned by *ERROR*.

**Parameters**

Parameter	Declaration	Data type	Memory block	Description
LADDR_IN / HW_ID	INPUT	WORD / HW_IO	I, Q, M, D, L	<ul style="list-style-type: none"> <li>■ LADDR_OUT <ul style="list-style-type: none"> <li>– Logical base input address of the frequency measurement module.</li> <li>– When used in CPUs from VIPA or in S7-300 CPUs from Siemens.</li> </ul> </li> <li>■ HW_ID <ul style="list-style-type: none"> <li>– <i>HW identifier</i> to address the frequency measurement module.</li> <li>– When used in S7-1500 CPUs from Siemens.</li> </ul> </li> </ul>
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)

Parameter	Declaration	Data type	Memory block	Description
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)
PERIOD_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Period duration
PERIOD_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Period duration

**LADDR\_IN**

Peripheral address:

- This parameter is available in CPUs from VIPA or in S7-300 CPUs from Siemens.
- Configured base address of the input area of the System SLIO frequency measurement module, which is to be read from. The address is hexadecimal.
- (Example: Address 100: *LADDR\_IN*: = W#16#64)

**HW\_ID**

HW identifier:

- This parameter is only available in S7-1500 CPUs of Siemens.
- Enter at *HW\_ID* the *HW identifier*, with which your module can be addressed accordingly. ↪ *Chapter 4.1 'HW identifier - HW\_ID' on page 10*

**DONE**

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

**PERIOD\_CHx**

Currently determined period duration of the corresponding channel in 100ns.

**ERROR (Return value)**

The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFFF
0x80E0	Channel 0: Determined number of edges = 0
0x80E1	Channel 1: Determined number of edges = 0
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFFFF
0x80E8	Channel 0: No valid measurement within the entered measurement period
0x80E9	Channel 1: No valid measurement within the entered measurement period



**Error of the internal called SFC 14**

Code	Description
0x808x	System error on the bus coupler
0x8090	<i>LADDR_IN</i> is not correct, possible reasons: <ul style="list-style-type: none"> <li>■ there is no module configured on this address</li> <li>■ limitation of the length of consistent data was not considered</li> <li>■ Basic address in parameter <i>LADDR_IN</i> was not entered in hexadecimal type</li> </ul>
0x8093	There is no bus coupler existing for <i>LADDR_IN</i> , to which consistent data can be written.
0x80A0	An access error was detected during peripheral access.
0x80B0	System error on the bus coupler
0x80B1	Specified length of the source area does not correspond to the configured user data length.
0x80B2	System error on the bus coupler
0x80B3	System error on the bus coupler
0x80C1	The data from the previous write request on the module are not processed by the module, yet.
0x80C2	System error on the bus coupler
0x80Fx	System error on the bus coupler
0x85xy	System error on the bus coupler
0x8xyy	General error information <a href="#">🔗 Chapter 4.2 'General and Specific Error Information RET_VAL' on page 10</a>

**5.1.4 FC 302 - FM\_GET\_FREQUENCY - Calculate frequency consistent****Description**

With the FC 302 FM\_GET\_FREQUENCY, you can calculate the frequency of the input signals of both channels of the System SLIO frequency measurement module. By this function internally SFC 14 - DPRD\_DAT for consistent reading of user data is called. Here, the error messages of the function block are returned by *ERROR*.

Parameters

Parameter	Declaration	Data type	Memory block	Description
LADDR_IN / HW_ID	INPUT	WORD / HW_IO	I, Q, M, D, L	<ul style="list-style-type: none"> <li>■ LADDR_IN                             <ul style="list-style-type: none"> <li>– Logical base input address of the frequency measurement module.</li> <li>– When used in CPUs from VIPA or in S7-300 CPUs from Siemens.</li> </ul> </li> <li>■ HW_ID                             <ul style="list-style-type: none"> <li>– <i>HW identifier</i> to address the frequency measurement module.</li> <li>– When used in S7-1500 CPUs from Siemens.</li> </ul> </li> </ul>
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)
FREQUENCY_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Frequency
FREQUENCY_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Frequency

**LADDR\_IN**

Peripheral address:

- This parameter is available in CPUs from VIPA or in S7-300 CPUs from Siemens.
- Configured base address of the input area of the System SLIO frequency measurement module, which is to be read from. The address is hexadecimal.
- (Example: Address 100: *LADDR\_IN* = W#16#64)

**HW\_ID**

HW identifier:

- This parameter is only available in S7-1500 CPUs of Siemens.
- Enter at *HW\_ID* the *HW identifier*, with which your module can be addressed accordingly. ↪ [Chapter 4.1 'HW identifier - HW\\_ID' on page 10](#)

**DONE**

Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

**FREQUENCY\_CHx**

Currently determined frequency of the corresponding channel in mHz.

**ERROR (Return value)**

The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80DA	Channel 0: Measured time value = 0

Code	Description
0x80DB	Channel 1: Measured time value = 0
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFFF
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFFFF
0x80E6	Channel 0: Frequency > 600kHz
0x80E7	Channel 1: Frequency > 600kHz
0x80E8	Channel 0: No valid measurement within the entered measurement period.
0x80E9	Channel 1: No valid measurement within the entered measurement period.

#### Error of the internal called SFC 14

Code	Description
0x808x	System error on the bus coupler
0x8090	<p><i>LADDR_IN</i> is not correct, possible reasons:</p> <ul style="list-style-type: none"> <li>■ there is no module configured on this address</li> <li>■ limitation of the length of consistent data was not considered</li> <li>■ Basic address in parameter <i>LADDR_IN</i> was not entered in hexadecimal type</li> </ul>
0x8093	There is no bus coupler existing for <i>LADDR_IN</i> , to which consistent data can be written.
0x80A0	An access error was detected during peripheral access.
0x80B0	System error on the bus coupler
0x80B1	Specified length of the source area does not correspond to the configured user data length.
0x80B2	System error on the bus coupler
0x80B3	System error on the bus coupler
0x80C1	The data from the previous write request on the module are not processed by the module, yet.
0x80C2	System error on the bus coupler
0x80Fx	System error on the bus coupler
0x85xy	System error on the bus coupler
0x8xyy	<p>General error information</p> <p>🔗 <i>Chapter 4.2 'General and Specific Error Information RET_VAL' on page 10</i></p>

### 5.1.5 FC 303 - FM\_GET\_SPEED - Calculate rotational speed consistent

#### Description

With the FC 303 FM\_GET\_SPEED, you can calculate the rotational speed of the input signals of both channels of the System SLIO frequency measurement module. By this function internally SFC 14 - DPRD\_DAT for consistent reading of user data is called. Here, the error messages of the function block are returned by *ERROR*.

#### Parameters

Parameter	Declaration	Data type	Memory block	Description
LADDR_IN / HW_ID	INPUT	WORD / HW_IO	I, Q, M, D, L	<ul style="list-style-type: none"> <li>■ LADDR_IN <ul style="list-style-type: none"> <li>– Logical base input address of the frequency measurement module.</li> <li>– When used in CPUs from VIPA or in S7-300 CPUs from Siemens.</li> </ul> </li> <li>■ HW_ID <ul style="list-style-type: none"> <li>– <i>HW identifier</i> to address the frequency measurement module.</li> <li>– When used in S7-1500 CPUs from Siemens.</li> </ul> </li> </ul>
RESOLUTION_CH0	INPUT	DINT	I, Q, M, D, L	Channel 0: Resolution of the sensor
RESOLUTION_CH1	INPUT	DINT	I, Q, M, D, L	Channel 1: Resolution of the sensor
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	return value (0 = OK)
SPEED_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Rotational speed
SPEED_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Rotational speed

#### LADDR\_IN

Peripheral address:

- This parameter is available in CPUs from VIPA or in S7-300 CPUs from Siemens.
- Configured base address of the input area of the System SLIO frequency measurement module, which is to be read from. The address is hexadecimal.
- (Example: Address 100: *LADDR\_IN*: = W#16#64)

#### HW\_ID

HW identifier:

- This parameter is only available in S7-1500 CPUs of Siemens.
- Enter at *HW\_ID* the *HW identifier*, with which your module can be addressed accordingly. ↪ *Chapter 4.1 'HW identifier - HW\_ID' on page 10*

#### RESOLUTION\_CHx

Enter here the resolution in increments per revolution for the corresponding channel .

- DONE** Ready signal of the function
- TRUE: Function was finished without error.
  - FALSE: Function is not active respectively there is an error.
- SPEED\_CHx** Currently determined rotational speed of the corresponding channel in revolutions per minute (rpm).
- ERROR (Return value)** The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80D6	Channel 0: Input value RESOLUTION_CH0 = 0
0x80D7	Channel 1: Input value RESOLUTION_CH1 = 0
0x80D8	Channel 0: Input value RESOLUTION_CH0 < 0
0x80D9	Channel 1: Input value RESOLUTION_CH1 < 0
0x80DA	Channel 0: Measured time value = 0
0x80DB	Channel 1: Measured time value = 0
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFFF
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFFFF
0x80E6	Channel 0: Determined rotational speed > max. (DINT)
0x80E7	Channel 1: Determined rotational speed > max. (DINT)
0x80E8	Channel 0: No valid measurement within the entered measurement period
0x80E9	Channel 1: No valid measurement within the entered measurement period

**Error of the internal called SFC 14**

Code	Description
0x808x	System error on the bus coupler
0x8090	<i>LADDR_IN</i> is not correct, possible reasons: <ul style="list-style-type: none"> <li>■ there is no module configured on this address</li> <li>■ limitation of the length of consistent data was not considered</li> <li>■ Basic address in parameter <i>LADDR_IN</i> was not entered in hexadecimal type</li> </ul>
0x8093	There is no bus coupler existing for <i>LADDR_IN</i> , to which consistent data can be written.
0x80A0	An access error was detected during peripheral access.
0x80B0	System error on the bus coupler
0x80B1	Specified length of the source area does not correspond to the configured user data length.
0x80B2	System error on the bus coupler
0x80B3	System error on the bus coupler
0x80C1	The data from the previous write request on the module are not processed by the module, yet.
0x80C2	System error on the bus coupler
0x80Fx	System error on the bus coupler
0x85xy	System error on the bus coupler
0x8xyy	General error information <a href="#">↗ Chapter 4.2 'General and Specific Error Information RET_VAL' on page 10</a>

**5.1.6 FC 310 ... 313 - Frequency measurement SLIO****Overview**

The following VIPA specific functions are used to control the System SLIO frequency measurement modules, if the consistency of the data are ensured by the bus protocol and consistent reading respectively writing with SFC 14 respectively SFC 15 is not possible. Within the functions there are "FM\_..." parameters, whose content is to be consistently connected to the corresponding input or output area of the frequency measurement module by means of the bus system. By calling the appropriate function the corresponding "FM\_..." parameters are automatically filled by the function.

Function	Symbol	Comment
FC 310	FM_CONTROL	Function to control the frequency measurement
FC 311	FM_CALC_PERIOD	Function to calculate the period duration
FC 312	FM_CALC_FREQUENCY	Function to calculate the frequency
FC 313	FM_CALC_SPEED	Function to calculate the rotational speed

### 5.1.7 FC 310 - FM\_CONTROL - Control frequency measurement

#### Description

The System SLIO Frequency measurement module is controlled by the FC 310 FM\_CONTROL. Since this FC does not internally call a block for consistent write access of data, you have to ensure consistent data transfer in your system.

#### Parameters

Parameter	Declaration	Data type	Memory block	Description
ENABLE_FM	INPUT	BOOL	I, Q, M, D, L	Enable frequency measurement
PRESET_CH0	INPUT	DINT	I, Q, M, D, L	Channel 0: Measurement period
PRESET_CH1	INPUT	DINT	I, Q, M, D, L	Channel 1: Measurement period
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	return value (0 = OK)
FM_PRESET_PERIOD_CH0	OUTPUT	DWORD	I, Q, M, D, L	Setpoint value for frequency measurement module output address: +0
FM_PRESET_PERIOD_CH1	OUTPUT	DWORD	I, Q, M, D, L	Setpoint value for frequency measurement module output address: +4
FM_CONTROL_CH0	OUTPUT	WORD	I, Q, M, D, L	Setpoint value for frequency measurement module output address: +8
FM_CONTROL_CH1	OUTPUT	WORD	I, Q, M, D, L	Setpoint value for frequency measurement module output address: +10

#### ENABLE\_FM

With setting *ENABLE\_FM* the corresponding CONTROL is generated and issued via *FM\_CONTROL\_CHx*. The measurement of both channels is started as soon as the content of *FM\_CONTROL\_CHx* was consistent transferred by the bus system to the frequency measurement module. The measurement of both channels is stopped by resetting *ENABLE\_FM*, after *FM\_CONTROL\_CHx* was consistent transferred to the frequency measurement module.



*Only as long as the frequency meters are started, evaluated values can be retrieved from the module. Otherwise you get the error message that the channels are disabled.*

**PRESET\_CHx** Enter here the measurement period in  $\mu\text{s}$  for the corresponding channel.  
Range of values:  $1\mu\text{s} \dots 8\,388\,607\mu\text{s}$

**DONE** Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

**FM\_PRESET\_PERIOD\_CHx** This parameter contains the measuring period for channel 0 respectively channel 1. The content is to be consistent connected with address +0 respectively +4 of the output area of the frequency measurement module, via the according bus system.

**FM\_CONTROL\_CHx** This parameter contains CONTROL, which is generated by *ENABLE\_FM*. The content for channel 0 respectively channel 1 is to be consistent connected with address +8 respectively +10 of the output area of the frequency measurement module, via the according bus system.

**ERROR (Return value)** The following code can be reported:

Code	Description
0x0000	No error
0x80D2	Channel 0: Input value measurement period $\leq 0$
0x80D3	Channel 1: Input value measurement period $\leq 0$
0x80D4	Channel 0: Input value measurement period $> 8\,388\,607\mu\text{s}$
0x80D5	Channel 1: Input value measurement period $> 8\,388\,607\mu\text{s}$

### 5.1.8 FC 311 - FM\_CALC\_PERIOD - Calculate period duration

**Description** With the FC 311 FM\_CALC\_PERIOD, you can calculate the period duration of the input signals of both channels. Since this FC does not internally call a block for consistent read access of data, you have to ensure consistent data transfer in your system.

#### Parameters

Parameter	Declaration	Data type	Memory block	Description
FM_PERIOD_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +0
FM_PERIOD_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +8



Parameter	Declaration	Data type	Memory block	Description
FM_RISING_EDGES_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +4
FM_RISING_EDGES_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +12
FM_STATUS_CH0	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +16
FM_STATUS_CH1	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +18
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)
PERIOD_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Period duration
PERIOD_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Period duration

**FM\_PERIOD\_CHx** This parameter contains the measured time value of channel 0 respectively channel 1. The content is to be consistent connected with address +0 respectively +4 of the input area of the frequency measurement module, via the according bus system.

**FM\_RISING\_EDGES\_CHx** This parameter contains the determined number of rising edges for channel 0 respectively channel 1. The content is to be consistent connected with address +8 respectively +12 of the input area of the frequency measurement module, via the according bus system.

**FM\_STATUS\_CHx** This parameter contains the status of channel 0 respectively channel 1. The content is to be consistent connected with address +16 respectively +18 of the input area of the frequency measurement module, via the according bus system.

**DONE** Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

**PERIOD\_CHx** Currently determined period duration of the corresponding channel in 100ns.

**ERROR (Return value)** The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFFF
0x80E0	Channel 0: Determined number of edges = 0
0x80E1	Channel 1: Determined number of edges = 0
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFFFF
0x80E8	Channel 0: No valid measurement within the entered measurement period
0x80E9	Channel 1: No valid measurement within the entered measurement period

### 5.1.9 FC 312 - FM\_CALC\_FREQUENCY - Calculate frequency

#### Description

With the FC 312 FM\_CALC\_FREQUENCY, you can calculate the period duration of the input signals of both channels. Since this FC does not internally call a block for consistent read access of data, you have to ensure consistent data transfer in your system.

#### Parameters

Parameter	Declaration	Data type	Memory block	Description
FM_PERIOD_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +0
FM_PERIOD_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +8
FM_RISING_EDGES_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +4
FM_RISING_EDGES_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +12

Parameter	Declaration	Data type	Memory block	Description
FM_STATUS_CH0	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +16
FM_STATUS_CH1	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +18
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)
FREQUENCY_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Calculated frequency
FREQUENCY_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Calculated frequency

**FM\_PERIOD\_CHx** This parameter contains the measured time value of channel 0 respectively channel 1. The content is to be consistent connected with address +0 respectively +4 of the input area of the frequency measurement module, via the according bus system.

**FM\_RISING\_EDGES\_CHx** This parameter contains the determined number of rising edges for channel 0 respectively channel 1. The content is to be consistent connected with address +8 respectively +12 of the input area of the frequency measurement module, via the according bus system.

**FM\_STATUS\_CHx** This parameter contains the status of channel 0 respectively channel 1. The content is to be consistent connected with address +16 respectively +18 of the input area of the frequency measurement module, via the according bus system.

**DONE** Ready signal of the function

- TRUE: Function was finished without error.
- FALSE: Function is not active respectively there is an error.

**FREQUENCY\_CHx** Currently determined frequency of the corresponding channel in mHz.

**ERROR (Return value)** The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80DA	Channel 0: Measured time value = 0

Code	Description
0x80DB	Channel 1: Measured time value = 0
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFFF
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFFFF
0x80E6	Channel 0: Frequency > 600kHz
0x80E7	Channel 1: Frequency > 600kHz
0x80E8	Channel 0: No valid measurement within the entered measurement period.
0x80E9	Channel 1: No valid measurement within the entered measurement period.

### 5.1.10 FC 313 - FM\_CALC\_SPEED - Calculate rotational speed

#### Description

With the FC 313 FM\_CALC\_SPEED, you can calculate the velocity of the input signals of both channels. Since this FC does not internally call a block for consistent read access of data, you have to ensure consistent data transfer in your system.

#### Parameters

Parameter	Declaration	Data type	Memory block	Description
FM_PERIOD_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +0
FM_PERIOD_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +8
FM_RISING_EDGES_CH0	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +4
FM_RISING_EDGES_CH1	INPUT	DWORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +12
FM_STATUS_CH0	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +16
FM_STATUS_CH1	INPUT	WORD	I, Q, M, D, L	Actual value of frequency measurement module input address: +18

Parameter	Declaration	Data type	Memory block	Description
RESOLUTION_CH0	INPUT	DINT	I, Q, M, D, L	Channel 0: Resolution of the sensor
RESOLUTION_CH1	INPUT	DINT	I, Q, M, D, L	Channel 1: Resolution of the sensor
DONE	OUTPUT	BOOL	I, Q, M, D, L	Ready signal (TRUE = OK)
ERROR	OUTPUT	WORD	I, Q, M, D, L	Return value (0 = OK)
SPEED_CH0	OUTPUT	DINT	I, Q, M, D, L	Channel 0: Calculated rotational speed
SPEED_CH1	OUTPUT	DINT	I, Q, M, D, L	Channel 1: Calculated rotational speed

- FM\_PERIOD\_CHx** This parameter contains the measured time value for channel 0 respectively channel 1. The content is to be consistent connected with address +0 respectively +4 of the input area of the frequency measurement module, via the according bus system.
- FM\_RISING\_EDGES\_CHx** This parameter contains the determined number of rising edges for channel 0 respectively channel 1. The content is to be consistent connected with address +8 respectively +12 of the input area of the frequency measurement module, via the according bus system.
- FM\_STATUS\_CHx** This parameter contains the status of channel 0 respectively channel 1. The content is to be consistent connected with address +16 respectively +18 of the input area of the frequency measurement module, via the according bus system.
- RESOLUTION\_CHx** Enter here the resolution in increments per revolution for the corresponding channel.
- DONE** Ready signal of the function
- TRUE: Function was finished without error.
  - FALSE: Function is not active respectively there is an error.
- SPEED\_CHx** Currently determined rotational speed of the corresponding channel in revolutions per minute (rpm).

**ERROR (Return value)** The following codes can be returned:

Code	Description
0x0000	No error
0x80D0	Channel 0 not in status active
0x80D1	Channel 1 not in status active
0x80D6	Channel 0: Input value RESOLUTION_CH0 = 0
0x80D7	Channel 1: Input value RESOLUTION_CH1 = 0
0x80D8	Channel 0: Input value RESOLUTION_CH0 < 0
0x80D9	Channel 1: Input value RESOLUTION_CH1 < 0
0x80DA	Channel 0: Measured time value = 0
0x80DB	Channel 1: Measured time value = 0
0x80DC	Channel 0: Measured time value < 0
0x80DD	Channel 1: Measured time value < 0
0x80DE	Channel 0: Measured time value > 0x7FFFFFFF
0x80DF	Channel 1: Measured time value > 0x7FFFFFFF
0x80E2	Channel 0: Determined number of edges < 0
0x80E3	Channel 1: Determined number of edges < 0
0x80E4	Channel 0: Determined number of edges > 0xFFFFFFFF
0x80E5	Channel 1: Determined number of edges > 0xFFFFFFFF
0x80E6	Channel 0: Determined rotational speed > max. (DINT)
0x80E7	Channel 1: Determined rotational speed > max. (DINT)
0x80E8	Channel 0: No valid measurement within the entered measurement period
0x80E9	Channel 1: No valid measurement within the entered measurement period

## 5.2 Energy Measurement

### 5.2.1 Overview



Please note that the blocks listed below are not included in the library for the Siemens TIA Portal for S7-1500 CPUs from Siemens.

#### 5.2.1.1 Terms

##### Measurand

A *measurand* is a physical quantity that can be measured such as current, voltage or temperature.

- 
-

<b>Measured value</b>	<p>A <i>measured value</i> is a value of a measurand, which is determined by measurement or by calculation.</p>
<b>ID</b>	<p>In the module each <i>measurand</i> one <i>ID</i> is assigned. The access to the measured value of a measurand happens by means of the corresponding <i>ID</i>.</p>
<b>DS-ID</b>	<p>As soon as the module is supplied by the DC 24V power section supply, the measurement is started and the counting of the energy counters is continued with the retentive stored counter values. The measured values of all the measurands are stored in the module with one record set ID <i>DS-ID</i>. The following must be observed:</p> <ul style="list-style-type: none"><li>■ All measured values with the same <i>DS-ID</i> come from the same measurement and are consistent.</li><li>■ By specifying the <i>DS-ID</i> you can address the individual measured values of the same measurement.</li><li>■ The <i>DS-ID</i> covers the values 1 ... 15.</li><li>■ To refresh the measured values the <i>DS-ID</i> is to be incremented by 1. The value 15 must be followed by 1.</li><li>■ If the <i>DS-ID</i> is incremented and there is still no new value available, the current value is returned with an error.</li><li>■ <i>DS-ID</i> = 0 - Auto increment mode<ul style="list-style-type: none"><li>– With <i>DS-ID</i> = 0 there is a request with <i>auto increment mode</i>. Here the module always returns the current measured value. As soon as a new measured value is available, here the <i>DS-ID</i> is incremented by one within the values 1 ... 15. If there is no new measured value available, the <i>DS-ID</i> is not changed and a error message is returned.</li></ul></li><li>■ The uniqueness of a measured value always consists of the <i>ID</i> of the measurand and the <i>DS-ID</i>.</li></ul>
<b>Frame</b>	<p>In the module you can combine some measurands to one data package (Frame), which is transferred in one step. One data package consists of 12byte user data. Considering the data length of 12 bytes, you can define the content of a frame by specifying the <i>ID</i> of the measurands. Up to 256 frames may be configured (<i>Frame 0 ... Frame 255</i>). The following must be observed:</p> <ul style="list-style-type: none"><li>■ The definition of <i>Frame 1</i> to <i>Frame 255</i> happens by the command <i>Set_Frame</i>.</li><li>■ <i>Frame 0</i> with the corresponding measurands can exclusively be specified by the parametrization.</li><li>■ With telegram type <i>Zero Frame</i> the data package of <i>Frame 0</i> can be accessed. After the start-up of the module there are automatic <i>Zero Frame</i> requests as long as the process data communication comes from the head module.</li></ul>
<b>FR-ID</b>	<p>When defining frames by means of '<i>Set Frame</i>', via the <i>FR-ID</i> these are assigned to a number between 0 ... 255. By specifying the <i>FR-ID</i> you can request the corresponding frame.</p>

**Data type**

In the following the data types are listed, which are used in the module. The length is to be considered particularly by the definition of *Frames*.

Data type	Length in byte	Description
UINT_8	1	Integer 8bit
UINT_16	2	Integer 16bit
UINT_32	4	Integer 32bit
INT_8	1	Signed integer 8bit
INT_16	2	Signed integer 16bit
INT_32	4	Signed integer 32bit
FLOAT	4	32bit floating point IEEE 754

**5.2.1.2 Functionality****Overview**

- The energy measuring module is used to measure the energy of a 3-phase connection. In addition to voltage, current and phase, the module determines many other measurands.
- Limit values can be parametrized for some measurands. When exceeding or falling below corresponding interrupt status bits are set. The module supports several commands (CMD). For example, interrupt status bits can be reset hereby.
- With the function block FB 325 and the associated data structure of type UDT 325, you can read energy measured values and interrupt status bits of the energy measurement module and commands can be executed on the module. In this case, the FB 325 communicates via the cyclic I/O data (16 bytes each) of the module, which must be specified accordingly when FB 325 is called.
- The real request interface is realized via the data structure of the type UDT 325. This makes simple control and evaluation possible, for example via a touch panel.

**Interconnection of the FB 325:**

- During the configuration, make sure that the parameters CHANNEL\_IN and CHANNEL\_OUT of the FB 325 are correctly interconnected. Otherwise, you will receive a timeout error message.
  - CHANNEL\_IN is to be interconnected to the 16byte input data of the energy measurement module.
  - CHANNEL\_OUT is to be interconnected to the 16byte output data of the energy measurement module.

**Cyclic measured value acquisition**

- By performing a manual reset after PowerON, you can avoid temporary error messages. To do this, you have to set bit 7 of the variable *Header.Control\_Global* in the data structure *MEAS\_DATA* of FB 325.
- With the basic settings of the UDT 325, all measured values of the energy measuring module are read with a period of 1s and stored in the data structure *MEAS\_DATA*. You can adjust the period via the variable *Header.Polltime* in the data structure *MEAS\_DATA* of the FB 325.

**Manual measured value acquisition**

For the manual measured value acquisition you have to set bit 1 of the variable *Header.Control\_Global* in the data structure *MEAS\_DATA* of the FB 325. If the bit is set, the measured values are read once by the energy measurement module and then the bit is reset.



**Selection of the *measurands***

By default, the measured values of all *measurands* are read periodically. However, you have the option of selecting the *measurands* in the data structure *MEAS\_DATA*. Via bit 0 of the variable *Data.[Name of the measurand].Read\_Mode* the access to the value of the corresponding *measurand* can be set. Please note that here the measurand IDs are grouped together. As soon as at least one measured value of a *measurand* of a group is to be read, the measured values of all *measurands* of this group are read. For example, if the value of the measurand with the ID no. 4 is to be read, so are those with ID no. 5 and 6 are read. There are the following groupings:

Group	IDs of the measurands	Group	IDs of the measurands	Group	IDs of the measurands
1	1, 2, 3	6	16, 17, 18	11	31, 32, 33
2	4, 5, 6	7	19, 20, 21	12	34, 35, 36
3	7, 8, 9	8	22, 23, 24	13	37, 38, 39
4	10, 11, 12	9	25, 26, 27	14	40, 41
5	13, 14, 15	10	28, 29, 30		

The measured values read are entered in the corresponding variables of *Data.[Name of the measurand].Value*. For unread measured values *Value* = 0.

**Command Interface**

You can trigger commands via the data structure *MEAS\_DATA* by setting the corresponding bits in the variable *Header.Cmd*. If several bits are set, they are sequentially processed. Here, the following commands are available:

- Bit 0: Reset all the energy counters
- Bit 1: Trigger reset on the current transformer
- Bit 2: Reset *status measurement*
- Bit 3: Writing the energy setpoints from "SetValues" to the ID3 ... ID8.

**Error behavior**

- Error messages that occur during the initialization of the block or when reading measured values can be found in the data structure *MEAS\_DATA* at *Header.Status\_Global*.
- Error messages that occur during command processing can be found at *Header.Status\_Cmd* and the detailed information at *Header.Error\_ID*
- In the event of an error, the function block continues the order processing. Here, the faulty jobs are repeated. The measured values in the data structure *MEAS\_DATA* are not affected by error messages.

### 5.2.2 FB 325 - EM\_COM\_R1 - Communication with 031-1PAxx

#### Overview



Please note that this block is not included in the library for Siemens TIA Portal for S7-1500 CPUs from Siemens.

This block enables the communication with the modules 031-1PAxx for energy metering and power measurement. For the communication a data block is necessary. Here the DB gets its structure from the UDT 325 EM\_DATA\_R1. The block has the following functionalities:

- Load default parameters after start-up
- Storage of parameters, limit values, measured values and messages
- Transfer of consistent measured values
- Writing set points
- Definition of the measured values by means of an UDT structure
- Communication by means of telegram type and ID
- Functional diagnostics, connection monitoring and error message evaluation

#### Parameter

Parameter	Declaration	Data type	Description
MODE	INPUT	BYTE	<ul style="list-style-type: none"> <li>■ 0x01 = Data exchange via process data</li> </ul> Currently only the MODE = 0x01 is supported
CHANNEL_IN	INPUT	ANY	Pointer to the input data <ul style="list-style-type: none"> <li>■ With MODE = 0x01 exclusively data type BYTE and length 16 are permitted.</li> </ul> Example: P#E100.0 BYTE 16 or P#DB10.DBX0.0 BYTE 16
CHANNEL_OUT	INPUT	ANY	Pointer to the output data <ul style="list-style-type: none"> <li>■ With MODE = 0x01 exclusively data type BYTE and length 16 are permitted.</li> </ul> Example: P#A100.0 BYTE 16 or P#DB10.DBX16.0 BYTE 16
MEAS_DATA	IN_OUT	UDT	<ul style="list-style-type: none"> <li>■ UDT for the measured values ↗ <i>Chapter 5.2.3 'UDT 325 - EM_DATA_R1 - Data structure for FB 325' on page 34</i></li> <li>■ Please note that this structure must not be in the temporary local data!</li> </ul>

### 5.2.3 UDT 325 - EM\_DATA\_R1 - Data structure for FB 325

#### 5.2.3.1 Structure

#### UDT 325



Please note that this block is not included in the library for Siemens TIA Portal for S7-1500 CPUs from Siemens.

The UDT 325 has a dynamic structure and has the following basic structure.

UDT areas	Description
UDT - Header	Structure for the header data
UDT - Data	Same data structure for the individual <i>measurands</i> . A <i>measurand</i> is a physical quantity that can be measured such as current, voltage and temperature. An overview of the measurands can be found in the manual for your energy measurement module.
...	
UDT - Data	
UDT - SetValue	Structure for the setpoint specification

UDT - Header	Declaration	Data type	Description
Timeout	INPUT	TIME	<ul style="list-style-type: none"> <li>Timeout for job processing. If <i>Timeout</i> is exceeded, the job is aborted and a corresponding error message is output.</li> </ul>
Polltime	INPUT	TIME	<ul style="list-style-type: none"> <li>Interval for the periodic reading</li> <li><i>Polltime</i> is only relevant if the measured values are periodically read in the interval of <i>Polltime</i>, i.e. if bit 0 of <i>Header.Control_Global</i> is set. If <i>Polltime</i> is less than the fastest possible interval, the measured values are read in the fastest possible interval.</li> </ul>
Control_Global	INPUT	BYTE	<p>0: de-activated, 1: activated</p> <ul style="list-style-type: none"> <li>Bit 0: Periodic execution according to the <i>Polltime</i> (default)</li> <li>Bit 1: Immediate execution - bit is reset after the execution.</li> <li>Bit 6 ... 2: reserved</li> <li>Bit 7: Re-initialization of the block by the configuration is sent again</li> </ul>
Status_Global	OUTPUT	BYTE	<p>Block status</p> <ul style="list-style-type: none"> <li>0x00: Not processed</li> <li>0x01: In process (BUSY)</li> <li>0x02: Ready without error (DONE)</li> <li>0x80: Error on processing (ERROR)</li> </ul>
Status Alarm_Global	OUTPUT	BYTE	<p>Corresponds to B3: Header byte 3 - <i>Common status</i></p> <ul style="list-style-type: none"> <li>Bit 0: Frequency <i>F_MAX</i> exceeded</li> <li>Bit 1: Frequency <i>F_MIN</i> undershot</li> <li>Bit 2: Temperature <i>T_MAX</i> exceeded</li> <li>Bit 3: Voltage <i>VRMS_MAX</i> exceeded</li> <li>Bit 4: Voltage <i>VRMS_MIN</i> undershot</li> <li>Bit 5: Efficiency <i>PF_MIN</i> undershot</li> <li>Bit 6: Current <i>IRMS_MAX</i> exceeded</li> <li>Bit 7: reserved</li> </ul>
Cmd	INPUT	BYTE	<p>0: de-activated, 1: activated</p> <ul style="list-style-type: none"> <li>Bit 0: Reset all the energy counters</li> <li>Bit 1: Trigger reset on the current transformer</li> <li>Bit 2: Reset <i>status measurement</i></li> <li>Bit 3: Writing the energy setpoints from "SetValue" to the ID3 ... ID8.</li> </ul> <p>If several bits are set, they are sequentially processed.</p> <p>Note: Writing of energy set points requires a protocol version major <math>\geq 1</math> and minor <math>\geq 1</math>!</p>

Energy Measurement &gt; UDT 325 - EM\_DATA\_R1 - Data structure for FB 325

UDT - Header	Declaration	Data type	Description
Status_Cmd	OUTPUT	BYTE	Status command <ul style="list-style-type: none"> <li>■ 0x00: Not processed</li> <li>■ 0x01: In process (BUSY)</li> <li>■ 0x02: Ready without error (DONE)</li> <li>■ 0x80: Error on processing (ERROR) - see ERROR_ID</li> </ul>
Jobtime	OUTPUT	TIME	■ Duration to read the measured values respectively to run a command.
DsID	OUTPUT	BYTE	Number of the current DS-ID ↳ 'DS-ID' on page 31
Frame_ID	OUTPUT	BYTE	Number of the current FR-ID ↳ 'FR-ID' on page 31
Error_ID	OUTPUT	WORD	Detailed error information
Status_ReadVersion	OUTPUT	BYTE	Status Read FW Version <ul style="list-style-type: none"> <li>■ 0x00 = never executed</li> <li>■ 0x01: Busy</li> <li>■ 0x02: Done</li> <li>■ 0x80: Error</li> </ul>
Reserved	STATIC	ARRAY of BYTE (1...15)	reserved
VersionInfo		Struct	The firmware version is determined automatically
FirmwareMajor	OUTPUT	Byte	Firmware version: Major
FirmwareMinor	OUTPUT	Byte	Firmware version: Minor
FirmwareRevision	OUTPUT	Byte	Firmware revision
ProtocollMajor	OUTPUT	Byte	Protocol version: Major
ProtocollMinor	OUTPUT	Byte	Protocol version: Minor
ProtocollRevsion	OUTPUT	Byte	Protocol version: Revision
ChipDateYear	OUTPUT	WORD	Date measuring chip: Year
ChipDateMonth	OUTPUT	Byte	Date measuring chip: Month
ChipDateDay	OUTPUT	Byte	Date measuring chip: Day

**Same data structure for the individual *measurands*. An overview of the *measurands* can be found in the manual for your energy measurement module.**

UDT - Data	Declaration	Data type	Description
Name	IN_OUT	STRUCT	■ Name of the <i>measurand</i>
Read_Mode	INPUT	BYTE	■ Bit 0: Accessing the measured value of the measurand <ul style="list-style-type: none"> <li>– 0: Measured value should not be read.</li> <li>– 1: Measured value should be read.</li> </ul>
Value	OUTPUT	DWORD	■ Current measured value

UDT - SetValue	Declaration	Data type	Description
SetValues		STRUCT	
EN_L1_CONSUMED	INPUT	DWORD	Setpoint active energy L1 consumer: UINT32, 1Wh

UDT - SetValue	Declaration	Data type	Description
EN_L1_DELIVERED	INPUT	DWORD	Setpoint active energy L1 producer: UINT32, 1Wh
EN_L2_CONSUMED	INPUT	DWORD	Setpoint active energy L2 consumer: UINT32, 1Wh
EN_L2_DELIVERED	INPUT	DWORD	Setpoint active energy L2 producer: UINT32, 1Wh
EN_L3_CONSUMED	INPUT	DWORD	Setpoint active energy L3 consumer: UINT32, 1Wh
EN_L3_DELIVERED	INPUT	DWORD	Setpoint active energy L3 producer: UINT32, 1Wh
EXCESS_ACTIVE_EN_CONSUME	INPUT	DWORD	Setpoint for overflow energy meter phase 1 ... 3 consumer <ul style="list-style-type: none"> <li>■ 0xXX112233 <ul style="list-style-type: none"> <li>– XX: not used</li> <li>– 11: Setpoint (byte) for overflow energy meter phase 1 consumer</li> <li>– 22: Setpoint (byte) for overflow energy meter phase 2 consumer</li> <li>– 33: Setpoint (byte) for overflow energy meter phase 3 consumer</li> </ul> </li> </ul> Is incremented by 1 in case of an overflow of the energy meter (ID = 1)
EXCESS_ACTIVE_EN_DELIVERED	INPUT	DWORD	Setpoint for overflow energy meter phase 1 ... 3 producer <ul style="list-style-type: none"> <li>■ 0xXX112233 <ul style="list-style-type: none"> <li>– XX: not used</li> <li>– 11: Setpoint (byte) for overflow energy meter phase 1 producer</li> <li>– 22: Setpoint (byte) for overflow energy meter phase 2 producer</li> <li>– 33: Setpoint (byte) for overflow energy meter phase 3 producer</li> </ul> </li> </ul> Is incremented by 1 in case of an overflow of the energy meter (ID = 2)

### 5.2.3.2 Error messages

ERROR ID	Description
0x0000	no error
0x800E	External error (no answer from Modbus)
0x8060	Error: A more recent protocol version is required
0x8070	Error: Parameter MODE
0x8073	Error: Parameter CHANNEL_IN does not match MODE
0x8074	Error: Parameter CHANNEL_OUT does not match MODE
0x8080	Error: 'Set Frame': Timeout detected during access
0x8081	Error: 'Read Frame': Timeout detected during access
0x8082	Error: 'CMD Frame': Timeout detected during access
0x8083	Error: Timeout when automatically reading the firmware information
0x8091	Error: Read measured value: Timeout detected when reading
0x80A1	Error: No newer record set exists
0x80A2	Error: 'DS-ID' mapping rule violated
0x80A3	Error: Telegram length

ERROR ID	Description
0x80A4	Error: Defined frame too big
0x80A5	Error: Requested frame not defined
0x80A6	Error: Requested date not available
0x80A7	Error: 'CMD Frame': Command could not be executed
0x80A8	Error: Invalid frame definition (SetFrame)
0x80A9	Error: Frame type unspecified (invalid request)
0x80AA	Error: Last parameter set was not valid
0x80AB	Error: Measuring module BUSY, no new data are transferred
0x80AF	Internal error - Please contact our hotline! On an internal error (0x0F) all the measurements are stopped and a reset of the module to default parameters is triggered! Here all counter values and frame configurations are deleted!

### 5.3 Motion Modules

#### 5.3.1 Overview

##### Blocks

The blocks listed below give you access to the System SLIO Motion modules:

- FB 320 - ACYC\_RW - Acyclic access to the System SLIO motion module
- FB 321 - ACYC\_DS - Acyclic parametrization System SLIO motion module
- UDT 321 - ACYC\_OBJECT-DATA - Data structure for FB 321

##### Supported motion modules

The following System SLIO motion modules are supported:

- 054-1BA00: FM 054 Motion Module - Stepper
- 054-1CB00: FM 054 Motion Module - 2xDC
- 054-1DA00: FM 054 Motion Module - Pulse Train RS422

##### Index - Subindex

The System SLIO motion module provides its data, such as "Profiling target position" via an object dictionary. In this object dictionary the objects are organized and addressable a unique number consisting of *Index* and *Subindex*. The number is specified as follows:

0x	Index (hexadecimal)	-	Subindex (decimal)
Example: 0x8400-03			



*To improve the structure and for expansion at System SLIO Motion Module another object numbering (index-assignment) is used besides the standard CiA 402.*

**Index - areas**

By separating into *Index* and *Subindex* a grouping is possible. The individual areas are divided into groups of related objects. This object dictionary is structured as follows for the System SLIO motion modules:

Index area	Content
0x1000 up to 0x6FFF	General data and system data
0x7000 up to 0x7FFF	Data of the digital input and output part
from 0x8000	Data of the axis or drives



*Information about the structure of the object dictionary can be found in the manual of your motion module.*



*Each object has a subindex 0. Calling an object with subindex 0, the number of available subindexes of the corresponding object is returned.*

**I/O address range**

The motion modules occupy a certain number of bytes in the I/O address area.

Head module	Backplane bus	Motion module	
CPU respectively bus coupler	→ ←	Process data	Acyclic channel

Via the *Acyclic channel* you can perform acyclic read and write commands. For this in the input/output area of the motion modules a data area for the acyclic communication was implemented. This area includes 8bytes output and 8bytes input data. When the blocks are used, communication takes place via the *Acyclic channel*.



*The data exchange with the motion module must be consistent over the length of the input or output data! It is recommended to control it via the process image. You can also use SFC 14 and 15 to consistently read and write the input or output data.*

**Interconnecting the FBs**

- During the configuration, make sure that the parameters CHANNEL\_IN and CHANNEL\_OUT of the FBs are correctly interconnected.
  - CHANNEL\_IN is to be interconnected to the input data of the *Acyclic channel* of the motion module.
  - CHANNEL\_OUT is to be interconnected to the output data of the *Acyclic channel* of the motion module.

Starting from the base address, the start address of the *Acyclic channel* for the input and output data can be reached via the following offset:

- 054-1BA00: FM 054 - Stepper: Base address + 26
- 054-1CB00: FM 054 - 2xDC: Base address + 50
- 054-1DA00: FM 054 - Pulse Train RS422: Base address + 26

**Example with base address 256:**

```
CHANNEL_IN    :=P#I 282.0 BYTE 10 // Base address 256 + 26
CHANNEL_OUT   :=P#Q 282.0 BYTE 10 // Base address 256 + 26
```



*Please note that you specify a length of 10byte, although the Acyclic channel internally uses 8byte!*

### 5.3.2 FB 320 - ACYC\_RW - Acyclic access to the System SLIO motion module

#### Description

With this block you can access the object dictionary of the System SLIO motion modules by means of your user program. Here the block uses an acyclic communication channel based on a request/response sequence. This is part of the input/output area of motion module.

The following System SLIO motion modules are supported:

- 054-1BA00: FM 054 - Stepper
- 054-1CB00: FM 054 - 2xDC
- 054-1DA00: FM 054 - Pulse Train RS422



*Due to the FB 321 internally calls the FB 320 and both blocks access the same database, for each channel (if multi-channel) you can use only one of these blocks in your user program! Also this block must be called per cycle only once!*



*The data exchange with the motion module must be consistent over the length of the input or output data! It is recommended to control it via the process image. You can also use SFC 14 and 15 to consistently read and write the input or output data.*

#### Parameters

Parameter	Declaration	Data type	Description
REQUEST	IN	BOOL	The job is started with edge 0-1.
MODE	IN	BYTE	Enter 0x01 for the acyclic protocol
COMMAND	IN	BYTE	0x11 = Reading a data object (max. 4byte) 0x21 = Writing a data object (max. 4byte)
INDEX	IN	WORD	Index of the object in the object dictionary - see the manual for the System SLIO motion module.
SUBINDEX	IN	BYTE	Subindex of the object dictionary - see the manual for the System SLIO motion module.
WRITE_LENGTH	IN	DINT	Length of the data to be written in byte (max. 4byte)
WRITE_DATA	IN	ANY	Pointer to the data to be written.
READ_DATA	IN	ANY	Pointer to the received data.



Parameter	Declaration	Data type	Description
CHANNEL_IN	IN	ANY	Pointer to the beginning of the acyclic channel in the input area of the motion module. Enter as length 10bytes. Examples P#I100.0 BYTE 10 or P#DB10.DBX0.0 BYTE 10
CHANNEL_OUT	IN	ANY	Pointer to the beginning of the acyclic channel in the output area of the motion module. Enter as length 10bytes. Examples P#Q100.0 BYTE 10 or P#DB10.DBX10.0 BYTE 10
READ_LENGTH	OUT	DINT	Length of the received data in byte. This value is to be rounded up to a multiple of 4, because the length specification is not transmitted.
DONE	OUT	BOOL	1: Job has been executed without error
BUSY	OUT	BOOL	0: There is no job being executed 1: Job is currently being executed
ERROR	OUT	BOOL	0: No Error 1: There is an error. The cause of the error is shown on the <i>ERROR_ID</i> parameter
ERROR_ID	OUT	WORD	Detailed error information



Please note that the parameters *WRITE\_DATA* and *READ\_DATA* are not checked for data type and length!

### Behavior of the block parameters

- Exclusiveness of the outputs
  - The outputs *BUSY*, *DONE* and *ERROR* are mutually exclusive. There can only one of these outputs be TRUE at the same time.
  - As soon as the input *REQUEST* is TRUE, one of the outputs must be TRUE.
- Output status
  - The outputs *DONE*, *ERROR*, *ERROR\_ID* and *READ\_LENGTH* are reset by an edge 1-0 at the input *REQUEST*, when the function block is not active (*BUSY* = FALSE).
  - An edge 1-0 at *REQUEST* does not affect the job processing.
  - If *REQUEST* is already reset during job processing, so it is guaranteed that one of the outputs is set at the end of the command for a PLC cycle. Only then the outputs are reset.
- Input parameter
  - The input parameters are taken with edge 0-1 at *REQUEST*. To change parameters, you have to trigger the job again.
  - If there is again an edge 0-1 at *REQUEST* during the job processing, an error is reported, no new command is activated and the answer rejected by the current command!
- Error handling
  - The block has 2 error outputs for displaying errors during order processing. *ERROR* indicates the error and *ERROR\_ID* shows an additional error number.
  - The outputs *DONE* and *READ\_LENGTH* designates a successful command execution and are not set when *ERROR* becomes TRUE.

- Behavior of the *DONE* output
  - The *DONE* output is set, when a command was successfully executed.
- Behavior of the *BUSY* output
  - The *BUSY* output indicates that the function block is active.
  - Busy is immediately set with edge 0-1 of *REQUEST* and will not be reset until the job was completed successfully or failed.
  - As long as *BUSY* is TRUE, the function block must be called cyclically to execute the command.




*If there is again an edge 0-1 at REQUEST during the job processing, an error is reported, no new command is activated and the answer rejected by the current command!*


## ERROR\_ID





ERROR_ID	Description
0x0000	There is no Error
0x8070	Faulty parameter <i>MODE</i>
0x8071	Faulty parameter <i>COMMAND</i>
0x8072	Parameter <i>WRITE_LENGTH</i> exceeds the maximum size
0x8073	Parameter <i>CHANNEL_IN</i> does not fit the parameter <i>MODE</i>
0x8074	Parameter <i>CHANNEL_OUT</i> does not fit the parameter <i>MODE</i>
0x8075	Impermissible command (edge 0-1 at <i>REQUEST</i> during job is executed)
0x8081	Error - read access - data do not exist Command rejected!
0x8091	Error - write access - data do not exist Command rejected!
0x8092	Error - write access - data out of range Command rejected!
0x8093	Error - write access - data can only be read Command rejected!
0x8094	Error - write access - data are write protected Command rejected!
0x8099	Error during acyclic communication Command rejected!

## Program structure

If no job is active, all output parameters must be set to 0. With an edge 0-1 at *REQUEST*, with the following approach a job is activated:

1.  Check if a job is already active, if necessary terminate job and output error.
  - ⇒ Check for *DONE* = 1 or *BUSY* = 0

2.  Interconnect the input parameters:
  - MODE
  - COMMAND
  - WRITE\_LENGTH
  - CHANNEL\_IN
  - CHANNEL\_OUT

⇒ Terminate job on error, otherwise continue with step 3.
3.  Save input parameters internally.
4.  Execute the desired command and wait until this has been carried out.
5.  Save and output the result of the command execution internally.
6.  Set all the output parameter to 0.

### 5.3.3 FB 321 - ACYC\_DS - Acyclic parametrization System SLIO motion module

#### Description

With this block you can parametrize you motion module motion module by means of your user program. Here you can store your parameters as *Object list* in a data block and transfer them via the acyclic communication channel in your motion module

The following System SLIO modules are supported:

- 054-1BA00: FM 054 motion module - Stepper
- 054-1CB00: FM 054 motion module - 2xDC
- 054-1DA00: FM 054 motion module - Pulse Train RS422



*Due to the FB 321 internally calls the FB 320 and both blocks access the same database, for each channel (if multi-channel) you can use only one of these blocks in your user program! Also this block must be called per cycle only once!*

#### Parameters

Parameter	Declaration	Data type	Description
REQUEST	IN	BOOL	The job is started with edge 0-1.
MODE	IN	BYTE	Enter 0x01 for the acyclic protocol.
READ_BACK	IN	BOOL	0: Written objects are not read back. 1: Written objects are read back immediately after the write operation and compared.
GROUP	IN	WORD	0x01...0x7F: Selection of a group in the object list. 0xFF: Section of all the objects in the object list.
OBJECT_DATA	IN	ANY	Pointer to the UDT. <a href="#">↪ Chapter 5.3.4 'UDT 321 - ACYC_OBJECT-DATA - Data structure for FB 321' on page 46</a>
CHANNEL_IN	IN	ANY	Pointer to the beginning of the acyclic channel in the input area of the motion module. Enter as length 10bytes. Examples P#I100.0 BYTE 10 or P#DB10.DBX0.0 BYTE 10

Parameter	Declaration	Data type	Description
CHANNEL_OUT	IN	ANY	Pointer to the beginning of the acyclic channel in the output area of the motion module. Enter as length 10bytes. Examples P#Q100.0 BYTE 10 or P#DB10.DBX10.0 BYTE 10
DONE	OUT	BOOL	1: Job has been executed without error.
BUSY	OUT	BOOL	0: There is no job being executed. 1: Job is currently being executed.
DATASET_INDEX	OUT	INT	Object that is currently being processed.
ERROR	OUT	BOOL	0: No Error 1: There is an error. The cause of the error is shown on the <i>ERROR_ID</i> parameter.
ERROR_ID	OUT	WORD	Detailed error information

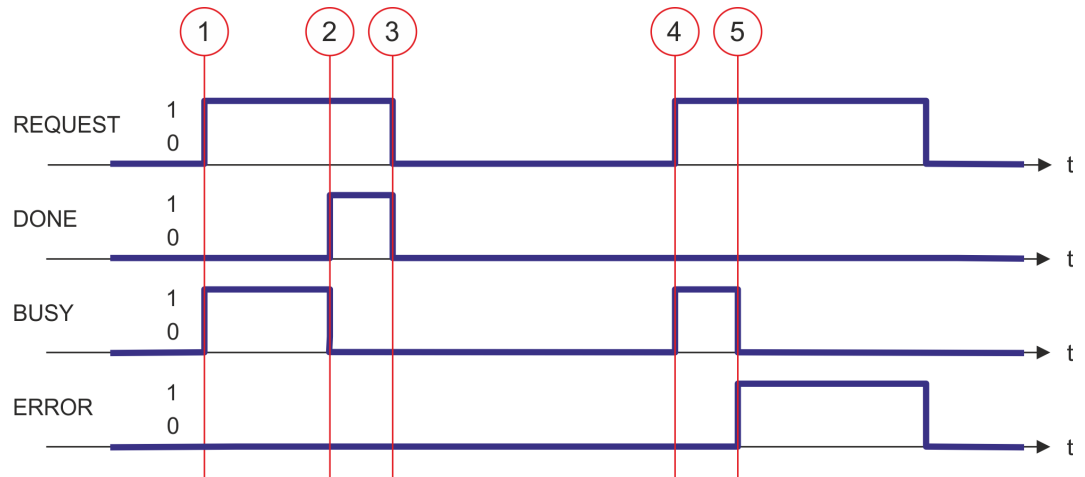
### Behavior of the block parameters

- Exclusiveness of the outputs:
  - The outputs *BUSY*, *DONE* and *ERROR* are mutually exclusive. There can only one of these outputs be TRUE at the same time.
  - As soon as the input *REQUEST* is TRUE, one of the outputs must be TRUE.
- Output status
  - The outputs *DONE*, *ERROR*, *ERROR\_ID* and *DATASET\_INDEX* are reset by an edge 1-0 at the input *REQUEST*, when the job is finished.
  - If *REQUEST* is already reset during job processing, so it is guaranteed that the whole object list is processed.
  - At the end of the job with no error, *DONE* is set for one PLC cycle. Only then the outputs are reset.
- Input parameter
  - The input parameters are taken with edge 0-1 at *REQUEST*. To change parameters, you have to trigger the job again.
  - If there is again an edge 0-1 at *REQUEST* during the job, an error is reported (invalid command sequence) and the processing of the object list is finished.
- Input parameter *READ\_BACK*
  - With activated parameter *READ\_BACK* written objects are read back immediately after the write operation by a read job.
  - The written and read values are compared.  
If they are identical, the next object is handled  
If they are not identical, an error message (*ERROR ID* = 0x8079) is returned and the development of the object list is finished.
- Input parameter *GROUP*
  - For a better structure you can assign a group to each object.
  - Via *GROUP* you define the group whose parameters are to be transferred.  
0x01...0x7F: Transfer the objects of the selected group.  
0xFF: Transfer the objects of all the groups.
- Error handling
  - The block has error outputs to show errors during job processing. *ERROR* indicates the error, *ERROR\_ID* shows an additional error number and *DATASET\_INDEX* informs at which object the error occurred.
  - The output *DONE* designates a successful job execution and is not set when *ERROR* becomes TRUE.
- Behavior of the *DONE* output
  - The *DONE* output is set, when a command was successfully executed.

- Behavior of the *BUSY* output
  - The *BUSY* output indicates that the function block is active.
  - *BUSY* is immediately set with edge 0-1 of *REQUEST* and will not be reset until the job was completed successfully or failed.
  - As long as *BUSY* is TRUE, the function block must be called cyclically to execute the command.
- Behavior of the *DATASET\_INDEX* output
  - The *DATASET\_INDEX* output indicates, which object of the object list is currently being processed.
  - If there is no job active, *DATASET\_INDEX* = 0 is returned.
  - If there is an error during the object processing, *DATASET\_INDEX* shows the faulting object.

**i** *If there is again an edge 0-1 at REQUEST during the job processing, an error is reported (ERROR\_ID = 0x8075), no new command is activated and the answer rejected by the current command!*

**Status diagram**



- (1) The job is started with edge 0-1 at *REQUEST* and *BUSY* becomes TRUE.
- (2) At the time (2) the job is completed. *BUSY* has the value FALSE and *DONE* den value TRUE.
- (3) At the time (3) the job is completed and *REQUEST* becomes FALSE and thus each output parameter FALSE respectively 0.
- (4) At the time (4) with an edge 0-1 at *REQUEST* the job is started again and *BUSY* becomes TRUE.
- (5) At the time (5) an error occurs during the job. *BUSY* has the value FALSE and *ERROR* den value TRUE.

**ERROR\_ID**

ERROR_ID	Description
0x0000	There is no Error
0x8070	Faulty parameter <i>MODE</i>
0x8071	Faulty parameter <i>OBJECT_DATA</i>
0x8075	Invalid command (edge 0-1 at <i>REQUEST</i> during job is executed)
0x8078	Faulty parameter <i>GROUP</i>
0x8079	<i>READ_BACK</i> detects an error (written and read value unequal)
0x807A	Pointer at <i>OBJECT_DATA</i> not valid



Within the function block the FB 320 is called. Here, any error of the FB 320 is passed to the FB 321. ↪ 'ERROR\_ID' on page 42

### 5.3.4 UDT 321 - ACYC\_OBJECT-DATA - Data structure for FB 321

#### Data structure for the object list

The parameters are to be stored in a data block as *object list*, which consists of individual *objects*. The structure of an *objects* is defined via an UDT.

#### Structure of an object

Variable	Declaration	Data type	Description
Group	IN	WORD	0 < Group < 0x80 permitted
COMMAND	IN	BYTE	0x11 = Read from the object list 0x21 = Write to the object list
Index	IN	WORD	Index of the object
Subindex	IN	BYTE	Subindex of the object
Write_Length	IN	BYTE	Length of the data to be written in byte
Data_Write	IN	DWORD	Data to be written.
Data_Read	OUT	DWORD	Read data
State	OUT	BYTE	0x00 = never processed 0x01 = <i>BUSY</i> - in progress 0x02 = <i>DONE</i> - successfully processed 0x80 = <i>ERROR</i> - an error has occurred during the processing



Please note that you always specify the appropriate length for the object during a write job!

#### Example DB

Addr.	Name	Type	Start value	Current value	Comment
0.0	Object(1).Group	WORD			1. Object
2.0	Object(1).Command	BYTE			
4.0	Object(1).Index	WORD			
6.0	Object(1).Subindex	BYTE			
7.0	Object(1).Write_Length	BYTE			
8.0	Object(1).Data_Write	DWORD			
12.0	Object(1).Data_Read	DWORD			
16.0	Object(1).State	BYTE			

Addr.	Name	Type	Start value	Current value	Comment
18.0	Object(2).Group	WORD			2. Object
...	...	...			
34.0	Object(2).State	BYTE			3. Object
36.0	Object(3).Group	WORD			
...	...	...			
52.0	Object(3).State	BYTE			...
...	...	...			

## 5.4 RAM to WLD - "WLD"

### 5.4.1 FB 240 - RAM\_to\_s7prog.wld - RAM to s7prog.wld

#### Description

With *REQ* = TRUE this block copies the currently loaded project of a CPU on an inserted memory card as s7prog.wld. With a SPEED7 CPU from VIPA the s7prog.wld is automatically read from an inserted memory card always after an overall reset. The FB 240 internally calls the block SFB 239 with the corresponding parameters. Here the values of *BUSY* and *RET\_VAL* are returned from the SFB 239 to the FB 240.



Please note that this block is not part of the library for the Siemens TIA Portal.

#### Parameters

Parameter	Declaration	Data type	Memory area	Description
REQ	IN	BOOL	I, Q, M, D, L	Function request with <i>REQ</i> = 1
BUSY	OUT	BOOL	I, Q, M, D, L	Return value of the SFB 239
RET_VAL	OUT	WORD	I, Q, M, D, L	Return value of the SFB 239

### 5.4.2 FB 241 - RAM\_to\_autoload.wld - RAM to autoload.wld

#### Description

With *REQ* = TRUE this block copies the currently loaded project of a CPU on an inserted memory card as autoload.wld. With a SPEED7 CPU from VIPA the s7prog.wld is automatically read from an inserted memory card always after PowerON. The FB 241 internally calls the block SFB 239 with the corresponding parameters. Here the values of *BUSY* and *RET\_VAL* are returned from the SFB 239 to the FB 241.



Please note that this block is not part of the library for the Siemens TIA Portal.

**Parameters**

Parameter	Declaration	Data type	Memory area	Description
REQ	IN	BOOL	I, Q, M, D, L	Function request with <i>REQ</i> = 1
BUSY	OUT	BOOL	I, Q, M, D, L	Return value of the SFB 239
RET_VAL	OUT	WORD	I, Q, M, D, L	Return value of the SFB 239

**5.5 Onboard I/O System 100V****5.5.1 SFC 223 - PWM - Pulse duration modulation****Description**

This block serves the parameterization of the pulse duration modulation for the last two output channels of X5.

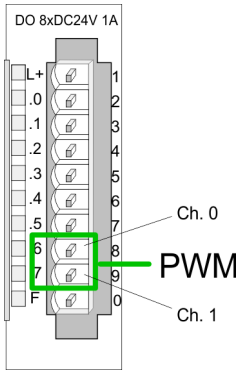


*Please note that this block is not part of the library for the Siemens TIA Portal.*

**Parameters**

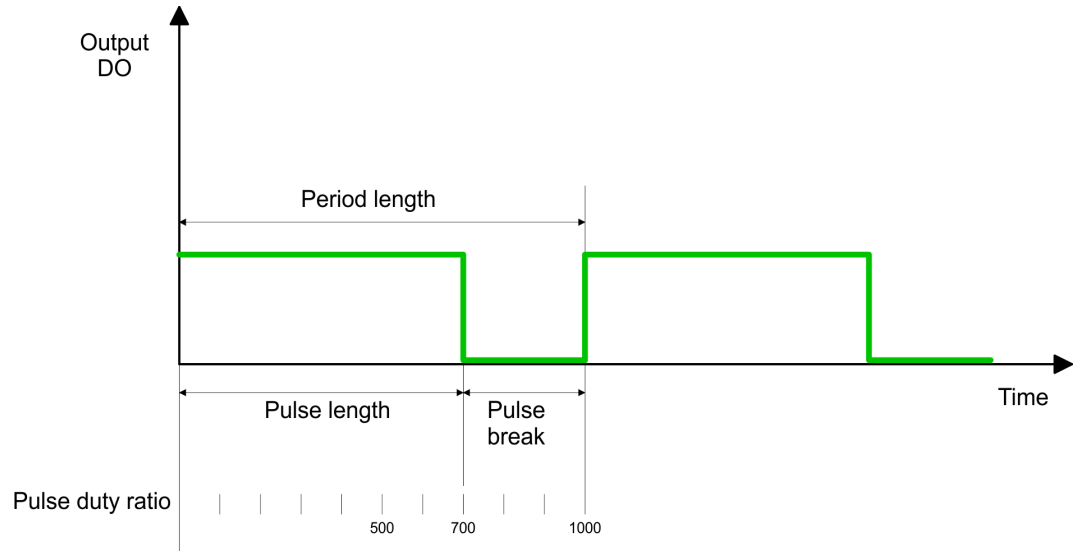
Parameter	Declaration	Type	Description
CHANNEL	IN	INT	Number of the output channel for PWM
ENABLE	IN	BOOL	Start bit of the job
TIMEBASE	IN	INT	Time base
PERIOD	IN	DINT	Period of the PWM
DUTY	IN	DINT	Output value per mille
MINLEN	IN	DINT	Minimum pulse duration
RET_VAL	OUT	WORD	Return value (0 = OK)





→ You define a time base, a period, the pulse duty ratio and min. pulse length. The CPU determines a pulse series with an according pulse/break relation and issues this via the according output channel.

⇒ The SFC returns a certain error code. You can see the concerning error messages in the table at the following page. The PWM parameters have the following relationship:



Period length = time base x period

Pulse length = (period length / 1000) x pulse duty ratio

Pulse break = period length - pulse length

The parameters have the following meaning:

**CHANNEL**

- Define the output channel that you want to address.
  - Value range: 0 ... 1

**ENABLE**

- Via this parameter you may activate the PWM function (true) res. deactivate it (false).
  - Value range: true, false

**TIMEBASE**

- *TIMEBASE* defines the resolution and the value range of the pulse, period and minimum pulse length per channel.
- You may choose the values 0 for 0.1ms and 1 for 1ms.
  - Value range: 0 ... 1

**PERIOD**

- Through multiplication of the value defined at period with the *TIMEBASE* you get the period length.
  - Value range: 0 ... 60000

- DUTY**
- This parameter shows the pulse duty ratio per mille. Here you define the relationship between pulse length and pulse break, concerned on one period.
    - 1 per mille = 1 *TIMEBASE*
  - If the calculated pulse duration is no multiplication of the *TIMEBASE*, it is rounded down to the next smaller *TIMEBASE* limit.
    - Value range: 0 ... 1000

- MINLEN**
- Via *MINLEN* you define the minimal pulse length. Switches are only made, if the pulse exceeds the here fixed minimum length.
    - Value range: 0 ... 60000

**RET\_VAL (Return Value)** Via the parameter *RET\_VAL* you get an error number in return. See the table below for the concerning error messages:

Value	Description
0000h	no error
8005h	Parameter <i>MINLEN</i> outside the permissible range
8006h	Parameter <i>DUTY</i> outside the permissible range
8007h	Parameter <i>PERIOD</i> outside the permissible range
8008h	Parameter <i>TIMEBASE</i> outside the permissible range
8009h	Parameter <i>CHANNEL</i> outside the permissible range.
9001h	Internal error - There was no valid address for a parameter.
9002h	Internal hardware error - Please contact the hotline.
9003h	Output is not configured as PWM output respectively there is an error in hardware configuration.
9004h	HF-PWM was configured but SFC 223 was called (please use SFC 225 HF_PWM!).

## 5.5.2 SFC 224 - HSC - High-speed-Counter

**Description** This SFC serves for parameterization of the counter functions (high speed counter) for the first 4 inputs.



*Please note that this block is not part of the library for the Siemens TIA Portal.*

### Parameters

Parameter	Declaration	Type	Description
CHANNEL	IN	INT	Number of the input channel for HSC
ENABLE	IN	BOOL	Start bit of the job
DIRECTION	IN	INT	Direction of counting

Parameter	Declaration	Type	Description
PRESETVALUE	IN	DINT	Preset value
LIMIT	IN	DINT	Limit for counting
RET_VAL	OUT	WORD	Return value (0 = OK)
SETCOUNTER	IN_OUT	BOOL	Load preset value

- CHANNEL**
- Type the input channel that you want to activate as counter.
    - Value range: 0 ... 3
- ENABLE**
- Via this parameter you may activate the counter (true) res. deactivate it (false).
    - Value range: true, false
- DIRECTION**
- Fix the counting direction.
    - Hereby is:
      - 0: Counter is deactivated, means *ENABLE* = false
      - 1: count up
      - 2: count down
- PRESETVALUE**
- Here you may preset a counter content, that is transferred to the according counter via *SETCOUNTER* = true.
    - Value range: 0 ... FFFFFFFFh
- LIMIT**
- Via Limit you fix an upper res. lower limit for the counting direction (up res. down). When the limit has been reached, the according counter is set zero and started new. If necessary an alarm occurs.
    - Value range: 0 ... FFFFFFFFh
- RET\_VAL (Return Value)**
- Via the parameter *RET\_VAL* you get an error number in return. See the table below for the concerning error messages:

Value	Description
0000h	No error
8002h	The chosen channel is not configured as counter (Error in the hardware configuration).
8008h	Parameter <i>DIRECTION</i> outside the permissible range
8009h	Parameter <i>CHANNEL</i> outside the permissible range
9001h	Internal error - There was no valid address for a parameter.
9002h	Internal hardware error - Please contact the hotline.

- SETCOUNTER**
- Per *SETCOUNTER* = true the value given by *PRESETVALUE* is transferred into the according counter.
  - The bit is set back from the SFC.
    - Value range: true, false

### 5.5.3 SFC 225 - HF\_PWM - HF pulse duration modulation

#### Description

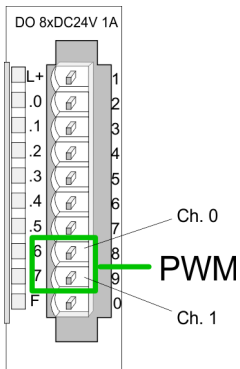
This block serves the parameterization of the pulse duration modulation for the last two output channels. This block is function identical to SFC 223. Instead of *TIMEBASE* and *PERIOD*, the SFC 225 works with a predefined frequency (up to 50kHz).



Please note that this block is not part of the library for the Siemens TIA Portal.

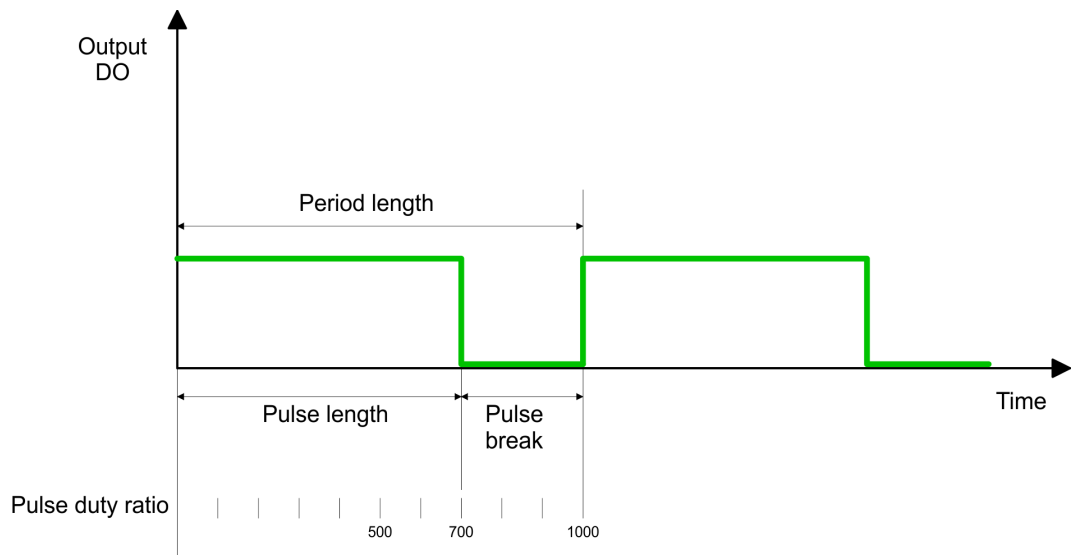
#### Parameters

Parameter	Declaration	Type	Description
CHANNEL	IN	INT	Number of the output channel for HF-PWM
ENABLE	IN	BOOL	Start bit of the job
FREQUENCY	IN	WORD	Frequency of the HF-PWM
DUTY	IN	DINT	Pulse duty ratio per mille
MINLEN	IN	DINT	Minimum pulse duration
RET_VAL	OUT	WORD	Return value (0 = OK)



→ You define a time base, a period, the pulse duty ratio and min. pulse length. The CPU determines a pulse series with an according pulse/break relation and issues this via the according output channel.

⇒ The SFC returns a certain error code. You can see the concerning error messages in the table at the following page. The PWM parameters have the following relationship:



$$\text{Period length} = 1 / \text{frequency}$$

$$\text{Pulse length} = (\text{period length} / 1000) \times \text{pulse duty ratio}$$

$$\text{Pulse break} = \text{period length} - \text{pulse length}$$

- CHANNEL**
- Define the output channel that you want to address.
    - Value range: 0 ... 1
- ENABLE**
- Via this parameter you may activate the PWM function (true) res. deactivate it (false).
    - Value range: true, false
- FREQUENCE**
- Type in the frequency in Hz as hexadecimal value.
    - Value range: 09C4h ... C350h (2.5kHz ... 50kHz)
- DUTY**
- This parameter shows the pulse duty ratio per mille. Here you define the relationship between pulse length and pulse break, concerned on one period.
    - 1 per mille = 1 *TIMEBASE*
  - If the calculated pulse duration is no multiplication of the *TIMEBASE*, it is rounded down to the next smaller *TIMEBASE* limit.
    - Value range: 0 ... 1000
- MINLEN**
- Via *MINLEN* you define the minimal pulse length in  $\mu\text{s}$ . Switches are only made, if the pulse exceeds the here fixed minimum length.
    - Value range: 0 ... 60000
- RET\_VAL (Return Value)** Via the parameter *RET\_VAL* you get an error number in return. See the table below for the concerning error messages:

Value	Description
0000h	no error
8005h	Parameter <i>MINLEN</i> outside the permissible range
8006h	Parameter <i>DUTY</i> outside the permissible range
8007h	Parameter <i>FREQUENCE</i> outside the permissible range
8008h	Parameter <i>TIMEBASE</i> outside the permissible range
8009h	Parameter <i>CHANNEL</i> outside the permissible range.
9001h	Internal error - There was no valid address for a parameter.
9002h	Internal hardware error - Please contact the hotline.
9003h	Output is not configured as PWM output respectively there is an error in hardware configuration.
9004h	HF-PWM was configured but SFC 223 was called (please use SFC 225 HF_PWM!).