

SIEMENS

SIMOTION Library for LMoMa modular machine




Application manual

<u>Preface</u>	1
<u>Library description</u>	2
<u>Library structure</u>	3
<u>Integration</u>	4
<u>Alarm and error messages</u>	5
<u>Examples and applications</u>	6
<u>Contacts and Internet addresses</u>	7

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.
 WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.
 CAUTION
with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.
CAUTION
without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.
NOTICE
indicates that an unintended result or situation can occur if the corresponding information is not taken into account.


If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation for the specific task, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

 WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be adhered to. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of the Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

1	Preface	7
1.1	General information	7
1.2	About this document	9
2	Library description	11
2.1	Description	11
2.1.1	General information	11
2.1.2	Overview of the functions of the LMoMa library.....	11
2.1.3	Topology view	13
2.2	Application range	14
2.3	Overview of automation and drive	15
2.3.1	Hardware structure	15
2.3.2	System requirements	15
2.3.3	Scope of delivery	16
3	Library structure.....	17
3.1	Structure of the library.....	17
3.2	FBs/FCs of the units from LMoMa	17
3.3	Graphic representation	20
3.4	fDOChanges unit	21
3.4.1	Task and functionality of the unit	21
3.4.2	FBLMoMaSetTopologyComparisonStage function block.....	22
3.4.2.1	General information	22
3.4.2.2	Schematic diagram in LAD	23
3.4.2.3	Input and output parameters.....	24
3.4.3	FBLMoMaChangeDriveCliqLineModule function block	25
3.4.3.1	General information	25
3.4.3.2	Schematic diagram in LAD	27
3.4.3.3	Input and output parameters.....	27
3.4.4	FBLMoMaChangeALMInfeedLineFilterType function block	28
3.4.4.1	General information	28
3.4.4.2	Schematic diagram in LAD	29
3.4.4.3	Input and output parameters.....	30
3.4.5	FBLMoMaChangeMotorModule function block.....	31
3.4.5.1	General information	31
3.4.5.2	Schematic diagram in LAD	33
3.4.5.3	Input and output parameters.....	34
3.4.6	FBLMoMaChangeMotorWithDriveCliq function block.....	35
3.4.6.1	General information	35
3.4.6.2	Schematic diagram in LAD	37
3.4.6.3	Input and output parameters.....	38
3.4.7	FBLMoMaChangeCatalogSMCMotor function block.....	39
3.4.7.1	General information	39

3.4.7.2	Schematic diagram in LAD.....	43
3.4.7.3	Input and output parameters.....	44
3.4.8	FBLMoMaChangeAsynchronousMotor function block.....	45
3.4.8.1	General information.....	45
3.4.8.2	Schematic diagram in LAD.....	47
3.4.8.3	Input and output parameters.....	47
3.4.9	FBLMoMaSetInhibitListForMotorCalculation function block.....	49
3.4.9.1	General information.....	49
3.4.9.2	Schematic diagram in LAD.....	50
3.4.9.3	Input and output parameters.....	51
3.4.10	FBLMoMaSetBrakeConfiguration function block.....	52
3.4.10.1	General information.....	52
3.4.10.2	Schematic diagram in LAD.....	53
3.4.10.3	Input and output parameters.....	54
3.4.11	FBLMoMaResetAndLoadParameterFromStorageMedium function block.....	55
3.4.11.1	General information.....	55
3.4.11.2	Schematic diagram in LAD.....	56
3.4.11.3	Input and output parameters.....	57
3.4.12	FBLMoMaCopySIMOTIONConfigDataToROM function block.....	58
3.4.12.1	General information.....	58
3.4.12.2	Schematic diagram in LAD.....	58
3.4.12.3	Input and output parameters.....	58
3.4.13	FBLMoMaSetActiveDDS function block.....	59
3.4.13.1	General information.....	59
3.4.13.2	Schematic diagram in LAD.....	60
3.4.13.3	Input and output parameters.....	60
3.4.14	FBLMoMaSetResetParkingAxis function block.....	62
3.4.14.1	General information.....	62
3.4.14.2	Schematic diagram in LAD.....	64
3.4.14.3	Input and output parameters.....	65
3.4.15	FBLMoMaActivateDeactivateSingleDOComponents function block.....	67
3.4.15.1	General information.....	67
3.4.15.2	Schematic diagram in LAD.....	68
3.4.15.3	Input and output parameters.....	68
3.4.16	FBLMoMaActivateDeactivateEncoder1AndSetActiveDDS function block.....	70
3.4.16.1	General information.....	70
3.4.16.2	Schematic diagram in LAD.....	70
3.4.16.3	Input and output parameters.....	71
3.4.17	FBLMoMaChangeTOConfigData function block.....	72
3.4.17.1	General information.....	72
3.4.17.2	Schematic diagram in LAD.....	73
3.4.17.3	Input and output parameters.....	74
3.4.18	FBLMoMaActivateDeactivateTOAndEncoder1 function block.....	76
3.4.18.1	General information.....	76
3.4.18.2	Schematic diagram in LAD.....	78
3.4.18.3	Input and output parameters.....	78
3.4.19	FBLMoMaAcceptNewEncoderSerialNumber function block.....	80
3.4.19.1	General information.....	80
3.4.19.2	Schematic diagram in LAD.....	80
3.4.19.3	Input and output parameters.....	81
3.5	Unit fRWTTopology.....	82
3.5.1	FBLMoMaReadDqTopology function block.....	82

3.5.1.1	General information	82
3.5.1.2	Schematic diagram in LAD	82
3.5.1.3	Input and output parameters.....	83
3.5.2	Function FCLMoMaGetAllInfosOfSingleComponent	87
3.5.2.1	General information	87
3.5.2.2	Schematic diagram in LAD	87
3.5.2.3	Input parameters	87
3.5.3	Function FCLMoMaCompareMLFB	88
3.5.3.1	General information	88
3.5.3.2	Schematic diagram in LAD	88
3.5.3.3	Input parameters	88
3.6	fTopology unit.....	89
3.6.1	FBLMoMaGetDOComponents function block.....	89
3.6.1.1	General information	89
3.6.1.2	Schematic diagram in LAD	89
3.6.1.3	Input and output parameters.....	90
3.6.2	FBLMoMaGetAllExistingDOsOfCU function block.....	93
3.6.2.1	General information	93
3.6.2.2	Schematic diagram in LAD	93
3.6.2.3	Input and output parameters.....	94
3.7	fDOIImage unit.....	95
3.7.1	General information	95
3.7.2	FBLMoMaHandleUnitData function block.....	95
3.7.2.1	General information	95
3.7.2.2	Schematic diagram in LAD	98
3.7.2.3	Input and output parameters.....	99
3.7.3	FBLMoMaSetDORamToRom function block.....	100
3.7.3.1	General information	100
3.7.3.2	Schematic diagram in LAD	100
3.7.3.3	Input and output parameters.....	101
3.7.4	FBLMoMaWriteBiCoToDO function block	102
3.7.4.1	General information	102
3.7.4.2	Schematic diagram in LAD	103
3.7.4.3	Input and output parameters.....	103
3.8	Constants	104
4	Integration.....	105
4.1	Integration in the SIMOTION project	105
4.2	Required technology objects.....	105
5	Alarm and error messages.....	107
5.1	Error messages.....	107
6	Examples and applications	113
6.1	Application example.....	113
7	Contacts and Internet addresses	125
7.1	Contacts	125
7.2	Internet addresses	125

Preface

1.1 General information

Note

The standard applications are not binding and do not claim to be complete regarding configuration, equipment or any eventuality which may arise. The standard applications do not represent specific customer solutions, but are only intended to provide support for typical tasks. You are responsible for the proper operation of the described products. These standard applications do not relieve you of your responsibility regarding the safe handling when using, installing operating and maintaining the equipment. By using these standard applications, you agree that Siemens cannot be made liable for possible damage beyond the mentioned liability clause. We reserve the right to make changes and revisions to these standard applications at any time without prior notice. In the case of any differences between the suggestions made in these standard applications and other publications from Siemens, such as catalogs, the contents of the other documentation have priority.

Warranty conditions, liability, and support

If the application has been made available free of charge, the following applies:

We do not provide a warranty for any of the information contained in this document.

All other rights and claims against Siemens AG irrespective of legal basis are excluded. In particular claims for damages against Siemens AG in the case of product outage, downtime, loss of profit, either directly, indirectly or consequential damage are excluded.

This does not apply when liability is compulsory by law, e. g. in the case of the Product Liability Act, premeditation, an act of gross negligence by superiors and managerial staff of Siemens AG or in cases of fraudulent concealment of defects.

This limitation of liability also applies to sub-contractors, suppliers, delegates, superiors and managerial staff of Siemens AG.

German law shall apply to this agreement for customers with head offices in Germany; Swiss law for customers with head offices outside Germany. Application of the United Nations Convention on Contracts for the International Sale of Goods as of 11.04.1980 (CISG) is excluded.

1.1 General information

If the application has been made available against payment, the appropriate alternative applies for the respective business transaction:

- Alternative 1: (Internal business)

If nothing else has been negotiated, then the "Conditions for the supply and services in Siemens internal business" applies in the version that is valid at the time that the equipment is purchased.

- Alternative 2: (Domestic business of Siemens AG)

If nothing else was negotiated, the "General License Conditions for Software for Automation and Drives for Customers with a Registered Office in Germany" valid at the time of sale are applicable.

- Alternative 3: (Direct export business of Siemens AG)

If nothing else has been negotiated, then the "General License Conditions for Software Products for Automation and Drives for Customers with a Seat or Registered Office outside Germany", valid at the time of sale, are applicable.

It is not permitted to distribute or duplicate these application examples in any form including excerpts thereof without the express consent of Siemens Industry Sector.

Notice regarding export identification codes

AL: N

ECCN: N

1.2 About this document

Aim

The function blocks and functions described in this document enable the user to manipulate the drive objects in SINAMICS via the application software in SIMOTION so that both variances occurring in the SINAMICS components used and variances in the quantity structure can be adapted during runtime.

In other words, during ramp-up, a SINAMICS S120 drive unit can be stopped and changes made to the drive objects with DPV1 services. For example, motors can be replaced without drawing on the help of an engineering tool such as STARTER. Furthermore, the components configured in the target topology and those currently installed can be read out and compared using the user software, right down to serial number level. This is relevant not only in the area of the pharmaceutical industry.

Target group

This document is intended for programmers and commissioning engineers.

Version history

Table 1- 1 Available versions

Date	Changes	Document version	Software revision
02/2011	Released document	1.03	1.0

Library description

2.1 Description

2.1.1 General information

This documentation describes functions (FC) and function blocks (FB) which can be used to read out the SINAMICS actual/target topology during SIMOTION system runtime.

It also describes FBs which enable SINAMICS drive objects to be manipulated during runtime and are necessary to execute typical applications.

All FBs are included in the **LMoMa** library. The names of the FB are, as far as possible, self-explanatory; see Section Library structure (Page 17).

Note

Visualization of the DRIVE-CLiQ topology on an HMI is not part of this library. However, the necessary data is available.

2.1.2 Overview of the functions of the LMoMa library

Possible modifications in the SIMOTION SCOUT project

Note

At this configuration level of the **LMoMa** library, the target topology stored on the SIMOTION device cannot be modified; changes can only be made at the drive object.

The following changes can be made using the functions of the **LMoMa** and **LDPV1** libraries.

2.1 Description

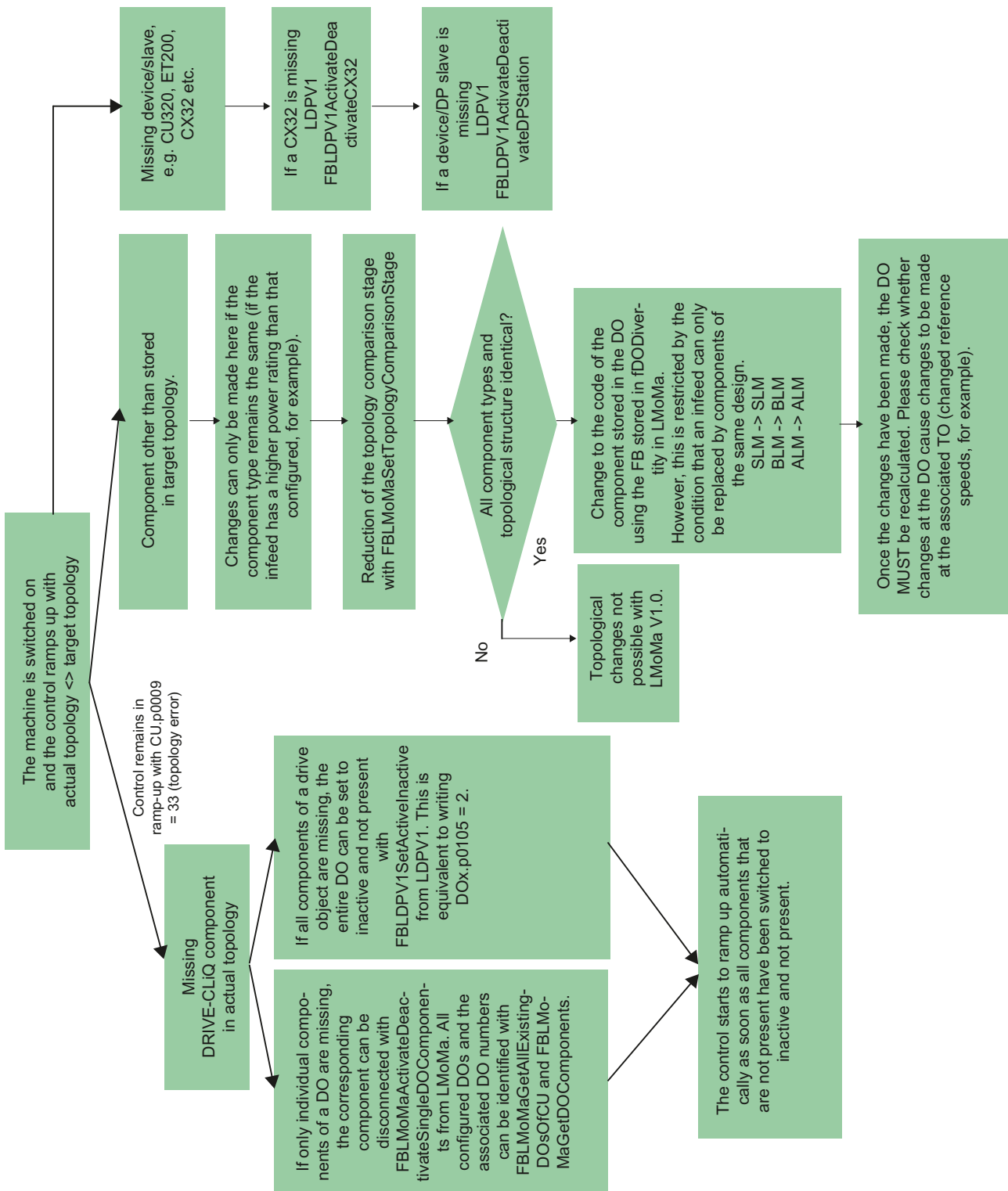


Figure 2-1 Possible modifications by the LMoMa library

2.1.3 Topology view

Topology view of all SINAMICS components contained in the SIMOTION SCOUT project

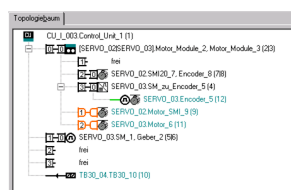


Figure 2-2 Topology view

The topology determines which components should be used or available (target/actual topology) and how these are connected together.

The following information is stored in the topology for each component:

- Type
- Device ID (MLFB)
- Serial number
- Wiring information (e.g. DRIVE-CLiQ)
- Versions of the firmware

A unique component number is also allocated for each component.

This component number is used to link a component with the associated drive object (DO). This information is available in the respective drive object.

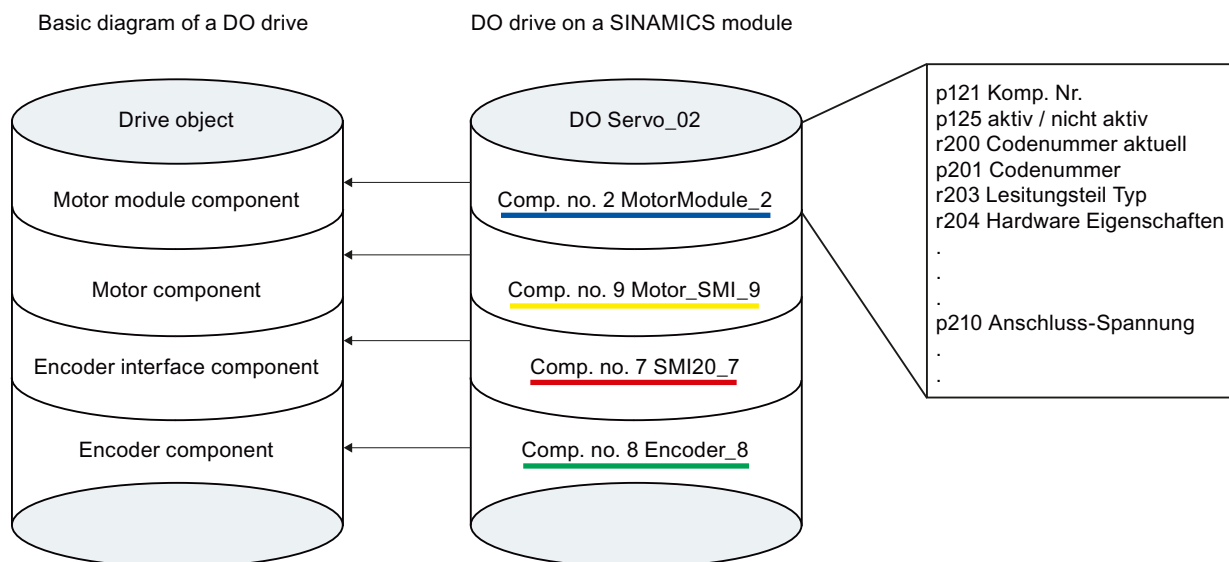


Figure 2-3 Representation of a DO drive

All data required for operating the relevant components is located in the drive object. It is stored in the corresponding parameters of the drive object and can be displayed and modified, for example, via the expert list.

To modify the operating data of the particular components at the drive object during runtime change from within the user program, you must use the function blocks of the **LMoMa** library from the **fDODiversity** unit. These make specific changes to the functional data and references of the individual components at the drive object.

2.2 Application range

The modular machine is the primary area of application of the **LMoMa** library.

The function blocks and functions implemented in the **LMoMa** library enable the configuration of a drive from the **SINAMICS family** stored on the storage medium (target topology) to be modified temporarily or permanently by a SIMOTION device during runtime so that components other than those configured can be used in the control cabinet or machine. This is particularly useful in the case of modular machines with variance (when parts are being replaced, for example) or even for the purpose of planned expansions.

2.3 Overview of automation and drive

2.3.1 Hardware structure

Hardware supported

The following hardware is supported:

- SIMOTION Integrated SINAMICS
- SIMOTION CX32 or SIMOTION CX32-2
- SINAMICS CU310
- SINAMICS CU320 or SINAMICS CU320-2

2.3.2 System requirements

The function blocks in the **LMoMa** library require the following versions to be used:

- SINAMICS Version V2.6.2
- SIMOTION SCOUT as of: Version V4.1.4

Note

Exception

SINAMICS drives configured with Drive Control Chart (DCC) are not supported by the SINAMICS **LMoMa** library.

The **LMoMa** library with safety has not yet been considered; it is to be tested separately.

Dependencies

Note

The **LMoMa** library can only be used in conjunction with the **LDPV1** library.

The library described in this document uses existing functionalities from Version V4.1 of the **LDPV1** library including buffer management, reading and writing of drive parameters, etc. As such, the use of the **LDPV1** library is a prerequisite for using the **LMoMa** library. Descriptions of the FBs and the functionalities of the **LDPV1** library can be found on the Utilities & Applications storage medium that is also part of SIMOTION SCOUT.

2.3.3 Scope of delivery

You will find the following data on the medium delivered:

- The **LMoMa** library in XML format

Library structure

3.1 Structure of the library

LMoMa library

The **LMoMa** library is organized into various units.

Table 3- 1 Units of the LMoMa library

Unit name	Application	Know-how protection
aVersion	Unit of the version overview, change list	No
cPublic	Unit of the definition of the constants that can be changed by the user	No
dProtected	Unit of the protected data	Yes
fChangesAtCU	No functionality is included in Version 1.0	Yes
fDODiversity	Function blocks for changing the functionality at the drive object	Yes
fDOIimage	Auxiliary functions for manipulating drive objects	Yes
fRWTopology	Functions and function blocks for reading topology information	Yes
fTopology	Auxiliary functions for additional information required to analyze the topology information read	Yes

3.2 FBs/FCs of the units from LMoMa

Units of the LMoMa library

The units described in this section are contained in the **LMoMa** library. No changes have to be made to the units by the user. The units are know-how-protected.

The complete topology, the actual topology, and the target topology stored on the storage medium of the SIMOTION device can be read out from SIMOTION during runtime with the function blocks of the fRWTopology unit from the **LMoMa** library.

aVersion unit

The *aVersion* unit of the **LMoMa** library documents the compatibility between the FBs and FCs and the software versions.

FBs of the fDODiversity unit

Table 3- 2 Function blocks of the fDODiversity unit

Function block	Integration/Call	Description
FBLMoMaSetTopologyComparisonStage	Can be called in all cyclic tasks	Lower the topology comparison stage
FBLMoMaChangeDriveCLiQLineModule	Can be called in all cyclic tasks	Changes the rating of the infeed used. The type of infeed must not be altered.
FBLMoMaChangeSingleMotorModule	Can be called in all cyclic tasks	
FBLMoMaChangeMotorModule	Can be called in all cyclic tasks	Makes changes to the motor module (SMM, DMM) at the drive object
FBLMoMaChangeMotorWithDriveCLiQ	Can be called in all cyclic tasks	Replaces the configured SMI motor with a different SMI motor
FBLMoMaChangeCatalogueSMCMotor	Can be called in all cyclic tasks	Replaces the configured catalog motor with SMC with a different catalog motor with identical SMC
FBLMoMaChangeAsynchronMotor	Can be called in all cyclic tasks	Replaces the configured asynchronous motor with a different asynchronous motor
FBLMoMaSetInhibitListForMotorCalculation	Can be called in all cyclic tasks	Creates a blocked list for certain parameters during recalculation of the drive
FBLMoMaSetBrakeConfiguration	Can be called in all cyclic tasks	Changes the configured setting for the holding brake
FBLMoMaResetAndLoadParameterFromStorageMedium	Can be called in all cyclic tasks	Resets a drive object to the configuration stored on the storage medium
FBLMoMaCopySIMOTIONConfigDataToROM	Can be called in a MotionTask	
FBLMoMaSetActiveDDS	Can be called in all cyclic tasks	
FBLMoMaSetResetParkingAxis	Can be called in all cyclic tasks	Activates/deactivates the parking axis on the drive
FBLMoMaActivateDeactivateSingleDOComponents	Can be called in all cyclic tasks	Explicitly activates/deactivates individual components of a drive object
FBLMoMaActivateDeactivateEncoder1AndSetActiveDDS	Can be called in all cyclic tasks	Changes over from a motor with encoder (e.g. DDS1) to a motor without encoder (e.g. DDS8)
FBLMoMaChangeTOConfigData	Can be called in all cyclic tasks	If changes are made to the DC, the necessary changes to the TO can be made here (e.g. reference speeds, gear ratio, etc.)
FBLMoMaSetActiveInactiveTOAndEncoder1	Can be called in all cyclic tasks	Hot-plugging of drives (only the motor is unplugged; the motor module remains plugged in)

Function block	Integration/Call	Description
FBLMoMaAcceptNewEncoderSerialNumber	Can be called in all cyclic tasks	Acknowledges the new serial number when replacing an encoder
FBLMoMaChangeALMInfeedLineFilterType	Can be called in all cyclic tasks	Adapts the line filter of an ALM

FBs/FCs of the fDOImage unit

Table 3- 3 Functions, and function blocks of the fDOImage unit

Function/Function block	Integration/Call
FBLMoMaHandleUnitData	Can be called in all cyclic tasks
FBLMoMaSetDORAMToROM	Can be called in all cyclic tasks
FBLMoMaWriteBiCoToDo	Can be called in all cyclic tasks

FBs/FCs of the fRWTopology unit

Table 3- 4 Functions and function blocks of the fRWTopology unit

Function/Function block	Integration/Call	Description
FBLMoMaReadDqTopology	Can be called in all cyclic tasks	Reads out the current actual or target topology
FCLMoMaGetAllInfosOfSingleComponent	Can be called in all tasks	Gets all the information for a component number from the transferred topology
FCLMoMaCompareMLFB	Can be called in all tasks	Two MLFB type STRINGS can be compared here.

To evaluate a topology, you need to know which DOs there are and which components belong to which DOs.

This information can be obtained during runtime using the function blocks provided in the *fTopology* unit of the **LMoMa** library.

FBs/FCs of the fTopology unit

Table 3- 5 Functions and function blocks of the fTopology unit

Function/Function block	Integration/Call	Description
FBLMoMaGetDOComponents	Can be called in all cyclic tasks	This FB identifies all component numbers belonging to a specific drive object (DO). The functions and function blocks necessary to make functional changes to individual components at the drive object have been implemented in the <i>fDODiversity</i> unit.
FBLMoMaGetAllExistingDOsOfCU	Can be called in all cyclic tasks	This FB identifies all drive objects (DO) existing on a SINAMICS device.

3.3 Graphic representation

Overview

The function blocks form a type of envelope around the call to SIMOTION functions and function blocks within the FBs in the **LDPV1** and **LMoMa** libraries. The error messages relating to system functions and system function blocks are evaluated accordingly and displayed externally to the user with special error messages which differ from the system error messages.

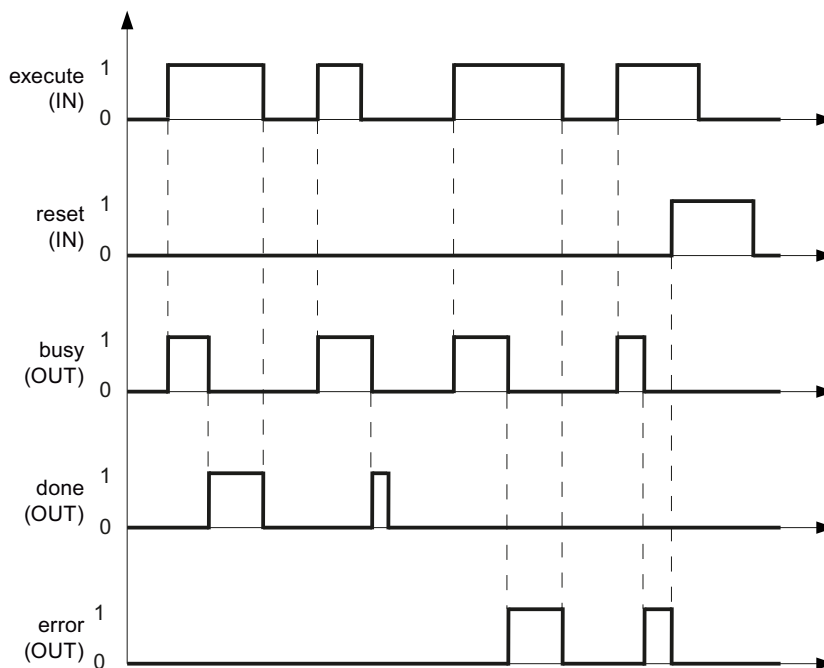


Figure 3-1 Time sequence diagram of the function block

Starting a function block

The respective function block is started with a rising edge at *execute*. On starting, the FB indicates that it is being actively processed by setting *busy* = *TRUE*. Once an FB has been processed, the *busy* reverts to *FALSE*. At the same time, the FB sets *done* = *TRUE* to signal that the functionality of the function block has been successfully completed.

This output state remains active until the user sets *execute* = *FALSE* to reset the FB to the initial state by initializing the outputs. Canceling *execute* before the FB is processed does not affect the sequence of the functionality. The FB sets *done* = *TRUE* for a cycle to signal that it has been processed. If *execute* is reset from *FALSE* to *TRUE* again before the functionality is complete, the current processing operation is aborted and restarted.

End of the function block with error

The function block is started with a rising edge at *execute*. On starting, the FB indicates that it is being actively processed by setting *busy = TRUE*. After this, an error affecting the processing of the FB is detected. This error is indicated by *error = TRUE* and a corresponding ID is displayed at the *errorID* output. This output state remains active until the user sets *execute = FALSE* to reset the FB to the initial state by initializing the outputs.

Resetting the function block with reset

The function block is reset with a rising edge at *reset*. On starting a reset, *busy = FALSE* is set immediately and active error messages are canceled. Communication jobs active within the processing of the FB are also terminated. After executing a reset, the *reset* input must be set to *FALSE* so that processing of the FB can start anew. If a rising edge at *execute* is detected during an active reset, a corresponding error is set. Reset is processed within a single processing cycle.

3.4 fDOChanges unit

3.4.1 Task and functionality of the unit

The function blocks implemented in the **fDOChanges** unit of the **LMoMa** library enable the configuration of a SINAMICS drive stored on the storage medium to be modified temporarily or permanently by a SIMOTION device so that components other than those configured can be used in the control cabinet or machine. This is particularly useful in the case of modular machines with variance (if the performance data of motor modules and motors changes, for example (5 A instead of 9 A)). The following function blocks are to be used exclusively on SINAMICS modules and deal with the changes necessary for variance at the relevant drive object.

Note

In the case of variance at the drive object, remember that changes at the configured drive object can affect dependencies between the drive object and the associated technology object (the reference speed of an axis ,for example). Changes at the technology object must be made independently by the user. The following variance blocks do not support such changes.

3.4.2 FBLMoMaSetTopologyComparisonStage function block

3.4.2.1 General information

Functionality

The **FBLMoMaSetTopologyComparisonStage** function block enables the user to change the topology comparison stage of a SINAMICS device during runtime. Changes can be stored temporarily or permanently (FBLMoMaSetDORAMToROM) in the SINAMICS component.

The comparison stage might need to be changed, for example, if a SINAMICS component starting up with topology errors terminates internal ramp-up. Depending on the set comparison stage, certain topology errors can be suppressed and the control units can terminate internal ramp-up. The following criteria are checked and evaluated during ramp-up for the individual comparison stages.

- High: Component type, order number, hardware version, manufacturer, and serial number
- Medium: Component type and order number
- Low: Component type
- Minimum: Component class

Procedure for changes to parameter p9906

Essentially, the function block writes the value of the comparison stage for all components to parameter *p9906* on the control unit of the device to be changed. However, since this parameter can only be changed under certain system conditions, the function block implements the following sequence.

- The device commissioning filter for the device is set to device configuration (*CU.p0009 = 1*). This is a must, as *p9906* can only be changed in this mode.
- The topology comparison stage to be set by input parameters is written to *CU.p9906*. Only the following values are permitted:
 - 0 = HIGH
 - 1 = MEDIUM
 - 2 = LOW
 - 3 = MINIMUM
- Once the topology comparison stage has been changed over, the value valid for the device commissioning filter before the start of the function block is reset to its original value before the FB was executed.
- If a topology error was active in the machine before the topology comparison stage was changed over, the new comparison stage is written to *CU.p9906* and the FB sets *done=TRUE* as soon as the ramp-up of the SINAMICS Integrated reaches *CU.p0002 =Ready to run* or **Operation**.
- Otherwise, the FB sets *done=TRUE* as soon as the reset to the original device parameter filter has been completed.

Acyclic communication

Internally, the function block uses acyclic communication DPV1 to write the parameters. For this reason, the function block must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for DPV1 services from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

3.4.2.2 Schematic diagram in LAD

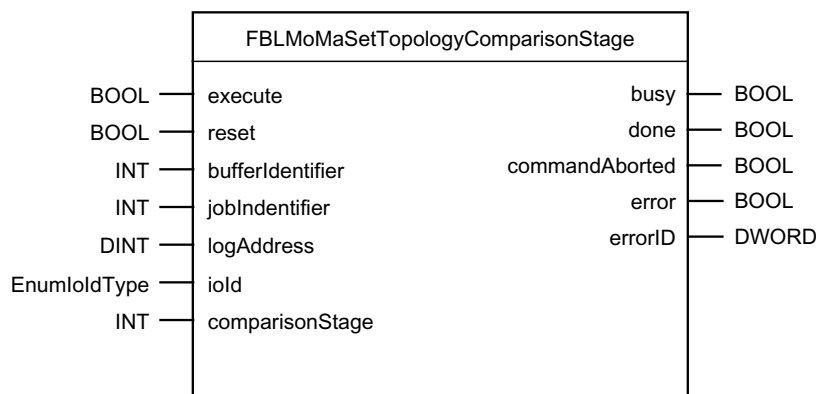


Figure 3-2 Schematic LAD diagram for FBLMoMaSetTopologyComparisonStage

3.4.2.3 Input and output parameters

Table 3- 6 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start status check/setting
reset	IN	BOOL	O	FALSE	Cancel function block
bufferIdentifier	IN	INT	O	-1	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the BuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of a SINAMICS drive object on which the access level is to be changed. Alternatively, the diagnostics address can be specified.
iold	IN	EnumIold type	O	INPUT	Specification of the data direction of the logical address (input/output)
comparisonStage	IN	INT	O	0	Specifies which topology comparison stage is to be set
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Function or job cancelled by a source external to the FB. Not relevant.
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.3 FBLMoMaChangeDriveCliqLineModule function block

3.4.3.1 General information

Functionality

The **FBLMoMaChangeDriveCliqLineModule** function block enables the configuration of an infeed stored on the storage medium to be modified temporarily or permanently (FBLMoMaSetDORAMToROM) by a SIMOTION device so that infeeds other than those configured can be used in the control cabinet or machine. This is particularly useful in the case of modular machines with variance. Only the first infeed can be changed in each case (index 0). No other infeeds (parallel connection) are taken into account at this software configuration level.

The following conditions must be observed:

- Only infeeds with DRIVE-CLiQ connection can be changed.
- An ALM can only be replaced by an ALM, an SLM by an SLM and a BLM replaced by a BLM.
- Changes from from ALM to SLM or BLM, SLM to ALM or BLM, and BLM to ALM or SLM are not possible.

Procedure for changes to parameter p9906

The FB has the following basic sequence:

- The currently connected infeed is read out (*DO.LM.r0200*)
- The current content of *DO.LM.p0573* is read out
- The code numbers from transfer parameters are compared with those read out (with potential abort if errors occur). If no code numbers are transferred to *moduleCode*, the infeed found in *DO.LM.r0200* is set automatically
- The infeed commissioning parameter filter is set to to power section commissioning (*DO.LM.p0010* = 2)
- *DO.LM.p0573* is set to indicate whether or not the infeed reference parameters (*DO.LM.p2000* – *DO.LM.p2006*) should be recalculated

- The infeed code number is written to the value of the new infeed (*DO.LM.p0201* = code number). The code number of an infeed can be obtained from its configuration screen form

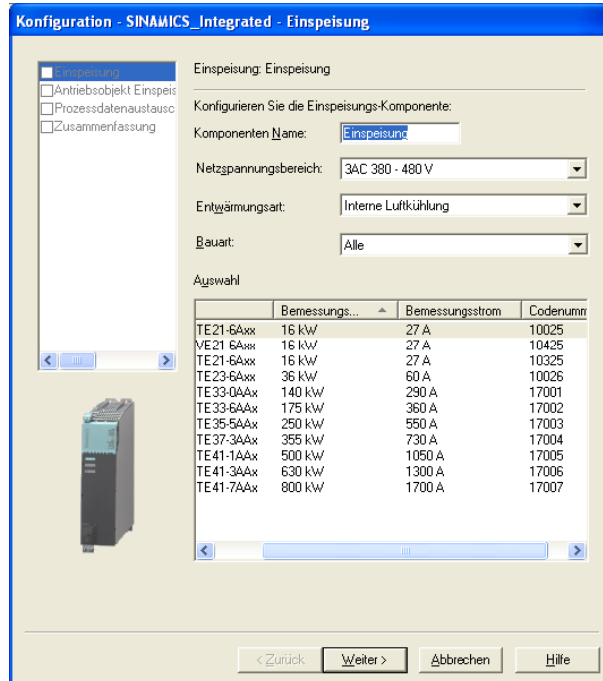


Figure 3-3 Module code numbers in the infeed configuration wizard

- The infeed commissioning parameter filter is set to quick commissioning (*DO.LM.p0010* = 1)
- Completion of quick commissioning (*DO.LM.p3900* = 3). Please see the documentation for parameter *p3900* for a description of what this involves
- Following recalculation, the old value for *DO.LM.p0573* is written back
- The FB signals **ready** as soon as parameter *DO.LM.p0010* signals **ready to run** again following completion of quick commissioning

Acyclic communication

Internally, the function block uses acyclic communication **DPV1** to write the parameters. For this reason, the FB must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for DPV1 services from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

3.4.3.2 Schematic diagram in LAD

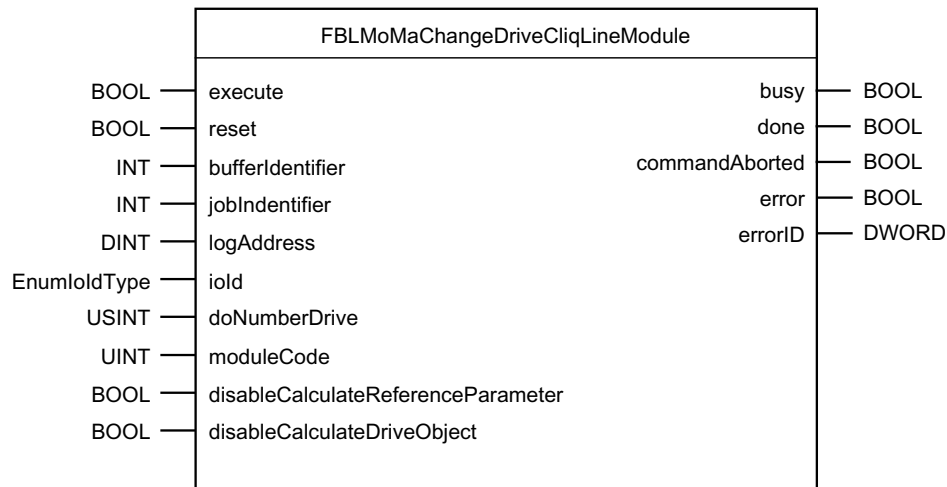


Figure 3-4 Schematic LAD diagram for FBLMoMaChangeDriveCliqLineModule

3.4.3.3 Input and output parameters

Table 3- 7 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start status check/setting
reset	IN	BOOL	O	FALSE	Cancel function block
bufferIdentifier	IN	INT	O	-1	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of a SINAMICS drive object on which the access level is to be changed. Alternatively, the diagnostics address can be specified.
iold	IN	EnumIold type	O	input	Specification of the data direction of the logical address (input/output)
doNumberDrive	IN	USINT	M	255	Drive object number of the infeed to be changed. If the drive object is transferred via the base address of the infeed in <i>logAddress</i> , <i>doNumberDrive</i> does not have to be supplied.
moduleCode	IN	UINT	M	0	Code number for infeed <i>p0201</i> of the infeed. If this value is 0, the infeed found in <i>r0200</i> is set automatically.
disableCalculateReferenceParameter	IN	BOOL	O	TRUE	Prevents the recalculation of the reference parameters from <i>p2000</i> through <i>p2006Do.LM.p0573</i> .

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
disableCalculatedDriveObject	IN	BOOL	O	FALSE	If the input is set to TRUE, recalculation of the drive object is not triggered
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Function or job cancelled by a source external to the FB. Not relevant.
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.4 FBLMoMaChangeALMInfeedLineFilterType function block

3.4.4.1 General information

Functionality

An ALM type power unit can be equipped with additional line filters. The **FBLMoMaChangeALMInfeedLineFilterType** function block can be used to change these line filters, including the optional basic line filter, during runtime.

The function block has the following basic sequence:

- The current content of *DO.LM.p0573* is read out
- The code numbers transferred for the required line filter are tested
- The infeed commissioning parameter filter is set to **quick commissioning** (*DO.ALM.p0010* = 1)
- *DO.ALM.p0573* is set to indicate whether or not the infeed reference parameters (*DO.ALM.p2000* – *DO.ALM.p2006*) should be recalculated
- The line filter code numbers are written to the values of the transferred values (*DO.ALM.p0220[0]* = line filter, *DO.ALM.p0220[1]* = optional BasicLineFilter). The code numbers for the line filters can be obtained from the parameter description for *p220*
- Completion of quick commissioning (*DO.ALM.p3900* = 3). Please see the documentation for parameter *p3900* for a description of what this involves

Following recalculation, the old value for *DO.ALM.p0573* is written back..

The FB signals **ready** as soon as parameter *DO.ALM.p0010* signals **ready to run** again following completion of quick commissioning.

Acyclic communication

Internally, the function block uses acyclic communication **DPV1** to write the parameters. For this reason, the FB must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for **DPV1** services from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

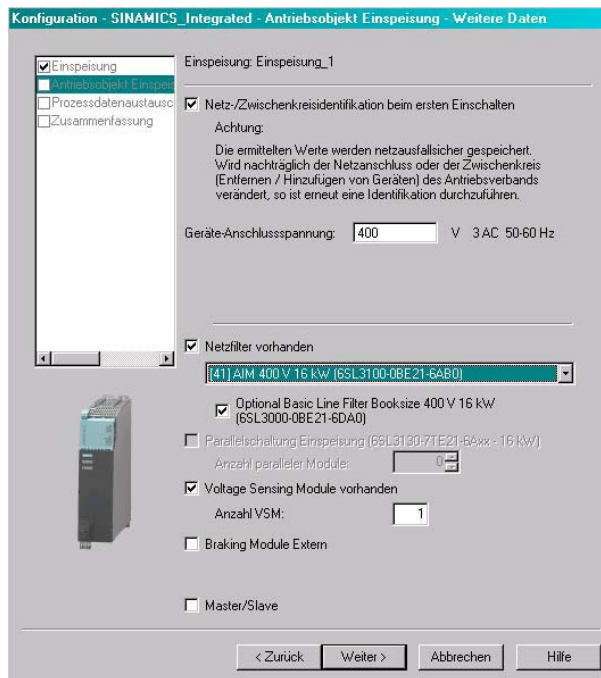


Figure 3-5 Configuration screen form for ALM with line filter

3.4.4.2 Schematic diagram in LAD

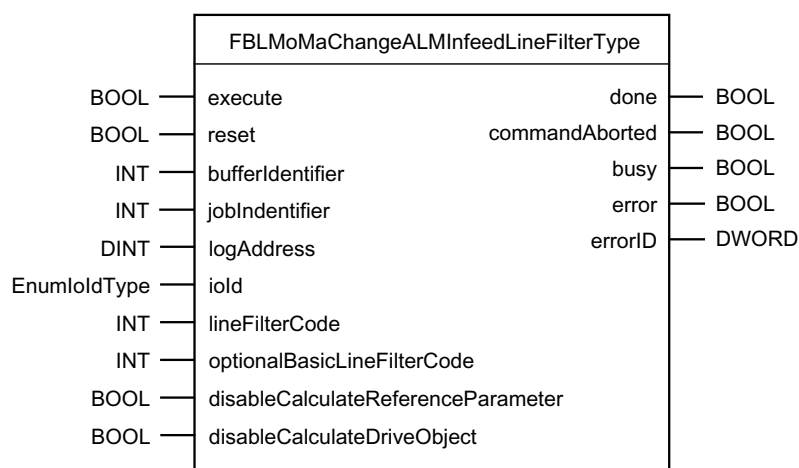


Figure 3-6 Schematic LAD diagram for FBLMoMaChangeALMInfeedLineFilterType

3.4.4.3 Input and output parameters

Table 3- 8 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop the function block
bufferIdentifier	IN	INT	O	-1	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	O	0	Any logical address of the SINAMICS on which the infeed is to be changed. Alternatively, the diagnostics address can be specified
ioId	IN	enumIoIdType	O	INPUT	Specification of the data direction of the transferred logical address (input/output)
doNumberDrive	IN	USINT	M	255	DO number of the infeed to be changed. If the drive object is transferred via the base address of the infeed in <i>logAddress</i> , <i>doNumberDrive</i> does not have to be supplied.
lineFilterCode	IN	INT	O	0	Line filter code for ALM to be set in <i>p220[0]</i>
optionalBasicLineFilterCode	IN	INT	O	0	Optional basic line filter code for ALM to be set in <i>p220[1]</i>
disableCalculateReferenceParameter	IN	BOOL	O	TRUE	Prevents the recalculation of the reference parameters from <i>p2000</i> through <i>p2006Do.LM.p0573</i> .
disableCalculateDriveObject	IN	BOOL	O	FALSE	If the input is set to TRUE, recalculation of the drive object is not triggered
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Processing cancelled by a source external to the FB
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.5 FBLMoMaChangeMotorModule function block

3.4.5.1 General information

Functionality

The **FBLMoMaChangeMotorModule** function block enables the configuration of a motor module stored on the storage medium to be modified temporarily or permanently (FBLMoMaSetDORAMToROM) by a SIMOTION device so that one parameterized motor module can be replaced by another in the control cabinet or machine.

The following conditions must be observed:

- A double motor module can only be replaced by another double motor module.
- A single motor module can be replaced by a single motor module.

It is not possible for a single motor module to be replaced by a double motor module and vice versa. A double motor module is always assigned to two motors. Therefore, a change to a double motor module must always be made to both associated motors. The FB makes the changes to the two motors. In each case, only the first motor module of the motors can be changed (index 0). No other motor modules (parallel connection) are taken into account at this software configuration level.

Procedure for changes to parameter p9906

The FB has the following basic sequence:

- The current content of *DO.Motor1.p0573* is read out
- The drive commissioning parameter filter is set to **quick commissioning** (*Do.Motor1.p0010* = 1)
- *Do.Motor1.p0573* is set to indicate whether or not the motor reference parameters (*Do.Motor1.p2000* – *Do.Motor1.p2006*) should be recalculated

- The motor module code number is written to the value of the new motor module (*Do.Motor1.p0201 = code number*). The code number of a motor module can be obtained from the corresponding drive's configuration screen form. If no code numbers are transferred to *moduleCode*, the infeed found in *DO.LM.r0200* is set automatically

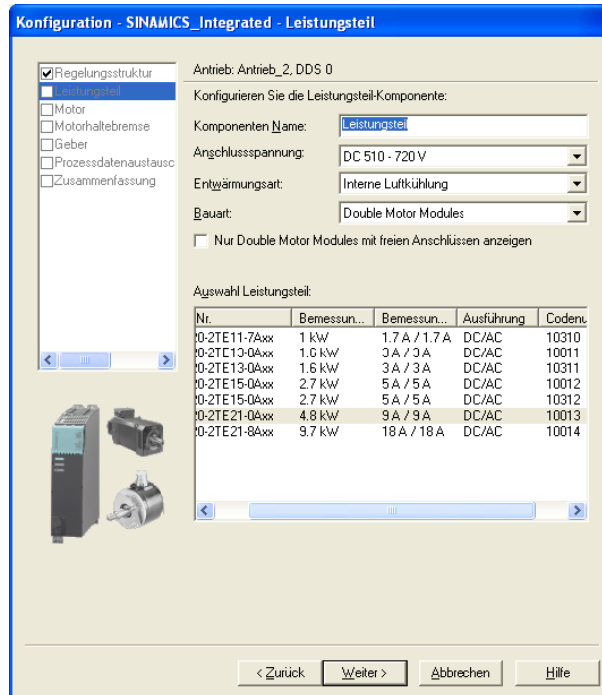


Figure 3-7 Motor module code numbers in the drive configuration wizard

- Completion of quick commissioning (*Do.Motor1.p3900 = 3*). Please see the documentation for parameter *p3900* for a description of what this involves. All the parameters of the associated motor are recalculated. If *disableCalculateReferenceParameter = TRUE*, the associated reference parameters are not recalculated. You can use the **FBLMoMaSetInhibitListForMotorCalculation** FB to exclude more motor parameters specifically from the recalculation
- Following recalculation, the old value for *DO.Motor1.p0573* is written back. The current content of *DO.Motor2.p0573* is read out. The drive commissioning parameter filter is set to **quick commissioning** (*Do.Antrieb2.p0010 = 1*)
- The motor module code number is written to the value of the new motor module. (*Do.Motor2.p0201 = code number*). The code number of a motor module can be obtained from the corresponding drive's configuration screen form
- Do.Motor2.p0573* is set to indicate whether or not the motor reference parameters (*Do.Motor2.p2000 – Do.Motor2.p2006*) should be recalculated

- Completion of quick commissioning (*Do.Motor2.p3900 = 3*). Please see the documentation for parameter *p3900* for a description of what this involves. All the parameters of the associated motor are recalculated. If *disableCalculateReferenceParameter = TRUE*, the associated reference parameters are not recalculated. You can use the **FBLMoMaSetInhibitListForMotorCalculation** FB to exclude more motor parameters specifically from the recalculation
- Following recalculation, the old value for *DO.Motor2.p0573* is written back
- The FB signals **ready** as soon as parameters *Do.Motor1.p0010* and *Do.Motor2.p0010* signal **ready to run** again following completion of quick commissioning

The code numbers for the motor modules to be transferred to the FB can be set in the configuration wizard of the associated drive.

Acyclic communication

Internally, the function block uses acyclic communication **DPV1** to write the parameters. For this reason, the FB must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for **DPV1 services** from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

3.4.5.2 Schematic diagram in LAD

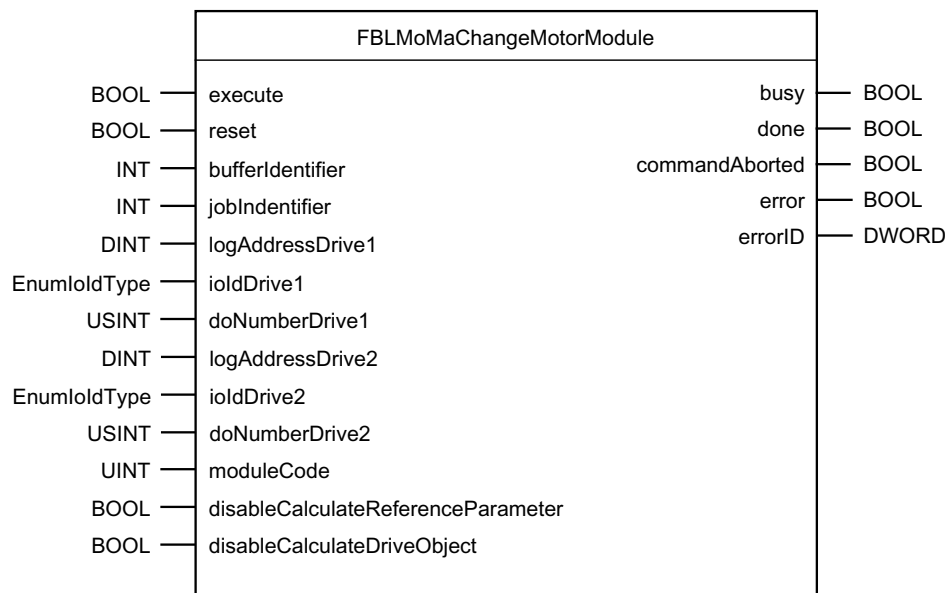


Figure 3-8 Schematic LAD diagram for FBLMoMaChangeMotorModule

3.4.5.3 Input and output parameters

Table 3-9 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start status check/setting
reset	IN	BOOL	O	FALSE	Cancel function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddressDrive1	IN	DINT	M	0	Any logical address of the SINAMICS drive on which the double/single motor module is to be changed. Alternatively, the diagnostics address can be specified.
ioldDrive1	IN	EnumIoId type	O	input	Specification of the data direction of the logical address (input/output)
doNumberDrive1	IN	USINT	M	255	Drive object number of the first motor to which the motor module is assigned. If the drive object is transferred via the base address of the drive in <i>logAddressDrive1</i> , <i>doNumberDrive1</i> does not have to be supplied.
logAddressDrive2	IN	DINT	M	0	Any logical address of the SINAMICS drive on which the double/single motor module is to be changed. Alternatively, the diagnostics address can be specified.
ioldDrive2	IN	EnumIoId type	O	input	Specification of the data direction of the logical address (input/output)
doNumberDrive2	IN	USINT	M	255	Drive object number of the second motor to which the motor module is assigned. If the drive object is transferred via the base address of the drive in <i>logAddressDrive2</i> , <i>doNumberDrive2</i> does not have to be supplied.
moduleCode	IN	UINT	M	0	Motor module code number. If this value is 0, the infeed found in <i>r0200</i> is set automatically.
disableCalculateReferenceParameter	IN	BOOL	O	TRUE	Prevents the recalculation of the reference parameters from <i>p2000</i> through <i>p2006Do.LM.p0573</i> .
disableCalculateDriveObject	IN	BOOL	O	FALSE	If the input is set to TRUE, recalculation of the drive object is not triggered

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Function or job cancelled by a source external to the FB. Not relevant.
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.6 FBLMoMaChangeMotorWithDriveCliq function block

3.4.6.1 General information

Functionality

The **FBLMoMaChangeMotorWithDriveCliq** function block enables the configuration of a motor with DRIVE-CLiQ interface stored on the storage medium to be modified temporarily or permanently (**FBLMoMaSetDORAMToROM**) by a SIMOTION device so that a detected motor with DRIVE-CLiQ interface can be read in again in the control cabinet or machine after being replaced by a different motor with DRIVE-CLiQ, for example. The encoder type is also read in again. The motor connected during this new detection process is then valid in the project. Motors in additional *DriveDataSets* are also taken into account. Motors with DRIVE-CLiQ interface can only be replaced by motors with DRIVE-CLiQ interface. The new motor and encoder are identified. Other settings such as the motor holding brake remain unaffected by this FB. The calculation of specific parameters can be suppressed with adjustable blocked lists. When replacing motors with/without brake, the following must be taken into account:

Table 3- 10 Instructions for replacing motors

Original motor	New motor	Comments
No brake	No brake	-
No brake	With brake	P1215 = motor holding brake as in sequence control (1)
With brake	No brake	Function is permitted. Parameter <i>p1215</i> should be re-parameterized, otherwise error message from drive
With brake	With brake	Settings are not changed

 **CAUTION**

For a motor with holding brake to be replaced by a motor without holding brake, the machine must be able to function in the same way with the new motor.

Procedure for changes to parameter p9906

The FB has the following basic sequence:

- The current content of *DO.Motor.p0180* is read out and the transferred DDS number is checked for validity
- The current content of *DO.Motor.p0573* is read out
- The MDS and EDS1 numbers belonging to the DDS number (*DO.Motor.p0186* and *DO.Motor.p0187*) are read out
- The motor commissioning parameter filter is set to **quick commissioning** (*Do.Motor.p0010* = 1)
- *Do.Motor.p0573* is set to indicate whether or not the motor reference parameters (*Do.Motor.p2000* – *Do.Motor1.p2006*) should be recalculated
- If set, the technological application of the motor is transferred (*Do.Motor.p0500*). The following values are permitted:
 - 100: Standard drive (servo)
 - 101: Feed drive (limit current limitation)
 - 102: Spindle drive (rated current limitation)
- The code numbers of the motor are written to the value for DRIVE-CLiQ motors (*Do.Motor.p0300[MDS]* = 10000 **motor identification**)
- The encoder code number is written to the value for DRIVE-CLiQ encoders (*Do.Motor.p0400[EDS1]* = 10000 **encoder identification**)

- Completion of quick commissioning (*Do.Motor.p3900 = 3*). Please see the documentation for parameter *p3900* for a description of what this involves. All the parameters of the associated motor are recalculated. If *disableCalculateReferenceParameter = TRUE*, the associated reference parameters are not recalculated. You can use the **FBLMoMaSetInhibitListForMotorCalculation** FB to exclude more motor parameters specifically from the recalculation. You should use it to block the recalculation of *p1520/p1521* if the reference parameters are also not to be changed, e.g. if parts are being replaced. Otherwise, torque reduction does not function with uniform scaling
- Following recalculation, the old value for *DO.Motor.p0573* is written back
- The FB signals **ready** as soon as parameter *Do.Antrieb1.p0010* signals **ready to run** again following completion of quick commissioning

Acyclic communication

Internally, the function block uses acyclic communication **DPV1** to write the parameters. For this reason, the FB must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for **DPV1 services** from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

3.4.6.2 Schematic diagram in LAD

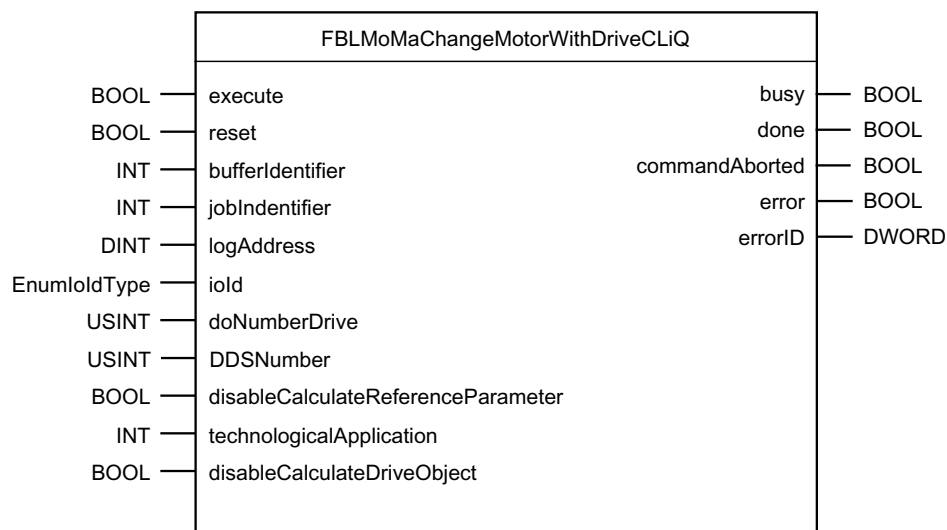


Figure 3-9 Schematic LAD diagram for FBLMoMaChangeMotorWithDriveCliq

3.4.6.3 Input and output parameters

Table 3- 11 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start status check/setting
reset	IN	BOOL	O	FALSE	Cancel function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of the SINAMICS drive on which the motor is to be changed. Alternatively, the diagnostics address can be specified.
iold	IN	EnumIold type	O	input	Specification of the data direction of the logical address (input/output)
doNumberDrive	IN	USINT	O	255	Drive object number of the motor whose DRIVE-CLiQ data is to be identified again. If the drive object is transferred via the base address of the drive in <i>logAddress</i> , <i>doNumber</i> does not have to be supplied.
DDSNumber	IN	UINT	M	0	Data set number for whose MDS and EDS1 the motor is to be changed
disableCalculateReferenceParameter	IN	BOOL	O	TRUE	Prevents the recalculation of the reference parameters from <i>p2000</i> through <i>p2006Do.LM.p0573</i> .
technologicalApplication	IN	INT	O	0	Technological application of the motor. The type determines which parameters are recalculated and how. If this value is not specified, the set technological application is retained (<i>Do.Motor.p0500</i>).
disableCalculateDriveObject	IN	BOOL	O	FALSE	If the input is set to TRUE, recalculation of the drive object is not triggered
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Function or job cancelled by a source external to the FB. Not relevant.
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

¹⁾ Parameter types: IN = input parameter, OUT = output parameter

²⁾ Parameter category: M = mandatory parameter, O = optional parameter

3.4.7 FBLMoMaChangeCatalogSMCMotor function block

3.4.7.1 General information

Functionality

The **FBLMoMaChangeCatalogSMCMotor** function block enables the configuration of a Siemens standard motor with SMC encoder stored on the storage medium to be modified temporarily or permanently (**FBLMoMaSetDORAMToROM**) by a SIMOTION device so that a parameterized standard motor with SMC encoder can be replaced in the control cabinet or machine by a different standard motor which also has an SMC encoder. The type of encoder can also be changed. The newly connected motor and the encoder are then valid in the project. Motors in additional *DriveDataSets* are also taken into account. If a new value is not specified for the motor or the encoder, the type of encoder or motor active thus far is retained. Motors with encoders that are connected to SMC modules can only be replaced by motors with encoders that are connected to SMC modules. The type of SMC may change, although this change cannot be made in the axis technology object online.

Note

The *_enableAxisInterface/_disableAxisInterface* system functions permit 2 axes to work with the same drive. If the encoder on the drive changes, these functions can be used to change the axis also.

Only the motor and the encoder can be set. Other settings such as the motor holding brake remain unaffected by this function block. If no encoder is present, or the encoder is not to be changed, no value is transferred in the associated parameter when the FB is called. If only the encoder is to be changed, no *MotorCode* input is required. Only motor and encoder codes listed in the SINAMICS List Manual are supported; i.e.:

Table 3- 12 Codes of motors

Motor code = P301	Motor type	Comments
0...9999	-	Error message
10200..10299 11200..11299 12200..12299	102	1PH2
10400..10499 11400..11499 12400..12499	104	1PH4
10700..10799 11700..11799 12700..12799	107	1PH7
10800..10899	108	1PH8
13400..13499 14400..14499 15400..15499	134	1PM4

Motor code = P301	Motor type	Comments
20600..20699 21600..21699 22600..22699	206	1FT6
26100..26199 26200..26299	261	1FE1
28300..28399 29300..29399	283	1FW3
28600..28699 29200..29299	286	1FW6
40300..40399 41300..41399	403	1FN3
Others	Motor code/100	If this is not known, error message when writing the parameter


Table 3- 13 Codes of motors

Encoder code	P4000	Comments
< 0	-	Error
0	-	Inactive/no change
1..9998	1..9998	If this is not known, error message when writing the parameter
< 9999	-	Error

When replacing motors with/without brake, the following must be taken into account:

Table 3- 14 Instructions for replacing motors

Original motor	New motor	Comments
No brake	No brake	-
No brake	With brake	P1215 = motor holding brake as in sequence control (1)
With brake	No brake	Function is permitted. Parameter <i>p1215</i> should be re-parameterized, otherwise error message from drive
With brake	With brake	Settings are not changed

 CAUTION
<p>For a motor with holding brake to be replaced by a motor without holding brake, the machine must be able to function in the same way with the new motor.</p>

Procedure for changes to parameter p9906

The FB has the following basic sequence:

- The current content of *DO.Motor.p0180* is read out and the transferred DDS number is checked for validity
- The current content of *DO.Motor.p0573* is read out
- The MDS and EDS1 numbers belonging to the DDS number (*DO.Motor.p0186* and *DO.Motor.p0187*) are read out
- The current encoder type is read out (*DO.Motor.P0400[EDS1]*)
- The current motor type is read out (*DO.Motor.P0301[MDS]*)
- The technological application is read out *DO.Motor.p0500*
- The motor commissioning parameter filter is set to **quick commissioning** (*Do.Motor.p0010* = 1)
- *DO.Motor.p0573* is set to indicate whether or not the motor reference parameters (*Do.Motor.p2000* – *Do.Motor1.p2006*) should be recalculated
- If set, the technological application of the motor is transferred (*Do.Motor.p0500*). The following values are permitted:
 - 100: Standard drive (servo)
 - 101: Feed drive (limit current limitation)
 - 102: Spindle drive (rated current limitation)
 If the technological application is not to be changed, the old value is written
- The code numbers of the motor are written to the value for DRIVE-CLiQ motors (*Do.Motor.p0300[MDS]* = 10000motor identification)

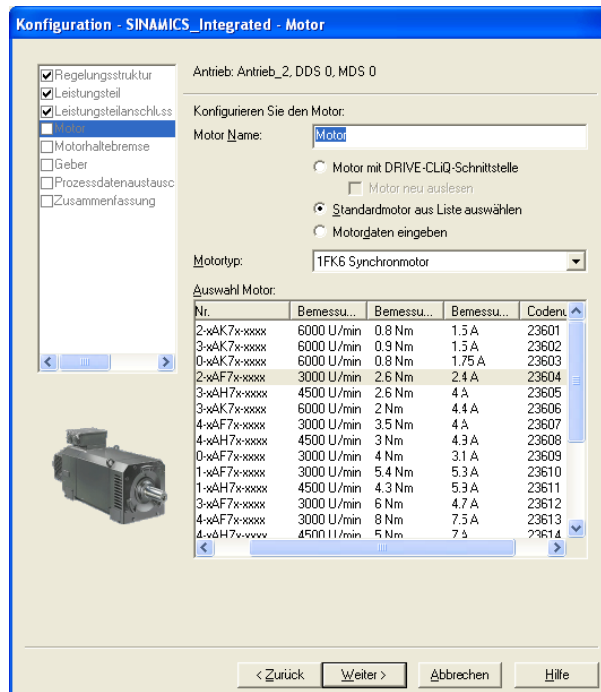


Figure 3-10 Motor code numbers in the drive configuration wizard

- The encoder code number is written to the value for DRIVE-CLiQ encoders (*Do.Motor.p0400[EDS1] = 10000 encoder identification*)

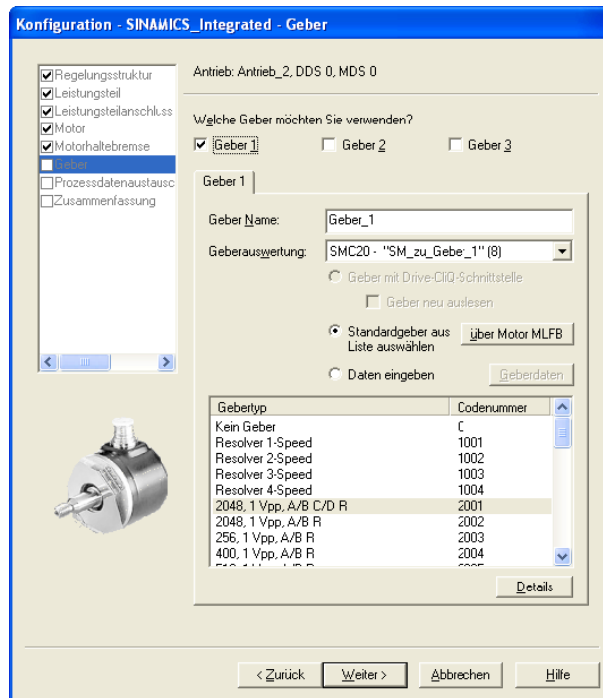


Figure 3-11 Encoder code numbers in the drive configuration wizard

- The motor code number is written to the (partial) value of the transfer parameter *motorCode* (*Do.Motor.p0300[MDS] = XXX first three digits of the parameter*). The code number of a motor can be obtained from the motor configuration screen form. If the motor is not to be changed, the old value is written
- The entire motor code number of the transfer parameter *motorCode* is written (*Do.Motor.p0301[MDS] = motorCode*). If the motor is not to be changed, the old value is written
- The encoder code number is written to the value from *encoderCode* (*Do.Motor.p0400[EDS1] = encoderCode*). The code number of an encoder can be obtained from a motor's configuration screen form. If the encoder is not to be changed, the old value is written
- Completion of quick commissioning (*Do.Motor.p3900 = 3*). Please see the documentation for parameter p3900 for a description of what this involves. All the parameters of the associated motor are recalculated. If *disableCalculateReferenceParameter = TRUE*, the associated reference parameters are not recalculated. You can use the **FBLMoMaSetInhibitListForMotorCalculation** FB to exclude more motor parameters specifically from the recalculation
- Following recalculation, the old value for *DO.Motor.p0573* is written back
- The FB signals **ready** as soon as parameter *Do.Antrieb1.p0010* signals **ready to run** again following completion of quick commissioning

Both the motor code number to be transferred to the FB and the encoder code number can be obtained from the configuration wizard of the associated drive.

Acyclic communication

Internally, the function block uses acyclic communication **DPV1** to write the parameters. For this reason, the FB must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for **DPV1 services** from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

3.4.7.2 Schematic diagram in LAD

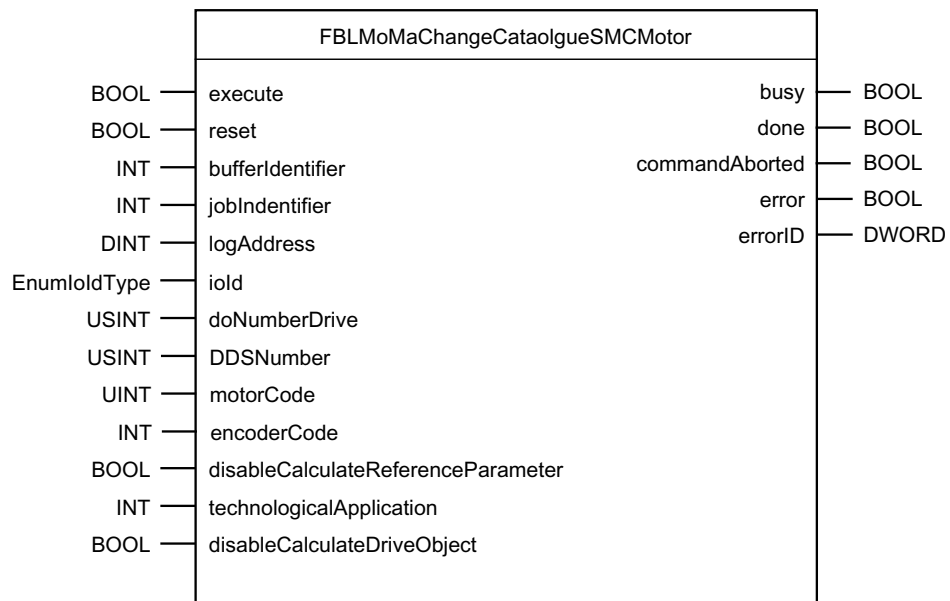


Figure 3-12 Schematic LAD diagram for FBLMoMaChangeCatalogSMCMotor

3.4.7.3 Input and output parameters

Table 3- 15 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start status check/setting
reset	IN	BOOL	O	FALSE	Cancel function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of the SINAMICS drive on which the motor is to be changed. Alternatively, the diagnostics address can be specified.
iold	IN	EnumIold type	O	input	Specification of the data direction of the logical address (input/output)
doNumberDrive	IN	USINT	O	255	Drive object number of the motor whose DRIVE-CLiQ data is to be identified again. If the drive object is transferred via the base address of the drive in <i>logAddress</i> , <i>doNumber</i> does not have to be supplied.
DDSNumber	IN	UINT	M	0	Data set number for whose MDS and EDS1 the motor is to be changed
motorCode	IN	UINT	O	0	Motor code number. Is the input is not set, the motor code does not change.
encoderCode	IN	INT	O	0	Encoder code number. Is the input is not set, the encoder does not change.
disableCalculateReferenceParameter	IN	BOOL	O	TRUE	Prevents the recalculation of the reference parameters from <i>p2000</i> through <i>p2006Do.LM.p0573</i>
disableCalculateDriveObject	IN	BOOL	O	FALSE	If the input is set to TRUE, recalculation of the drive object is not triggered
technologicalApplication	IN	INT	O	0	Technological application of the motor. The type determines which parameters are recalculated and how. If this value is not specified, the set technological application is retained (<i>Do.Motor.p0500</i>).
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.8 FBLMoMaChangeAsynchronousMotor function block

3.4.8.1 General information

Functionality

The **FBLMoMaChangeAsynchronousMotor** function block enables the configuration of an asynchronous motor stored on the storage medium to be modified temporarily or permanently (**FBLMoMaSetDORAMToROM**) by a SIMOTION device so that a parameterized asynchronous motor can be replaced in the control cabinet or machine by a different asynchronous motor. The motor is specified by entering its motor-specific data (as indicated on its type plate). The newly connected motor is then valid in the project. It is only possible for an asynchronous motor without encoder to be replaced by an asynchronous motor without encoder or an asynchronous motor with encoder to be replaced by an asynchronous motor with encoder. Motors in additional *DriveDataSets* are also taken into account. Asynchronous motors can only be replaced by other asynchronous motors. External servomotors are not supported. All data required by the FB in the input structure for the motor must be entered. Motors without encoders can only be replaced by motors without encoders. Similarly, motors with encoders can only be replaced by motors with encoders. If no new encoder code is transferred to the FB, the set encoder (if there is one) remains unaffected.

Procedure for changes to parameter p9906

The FB has the following basic sequence:

- The current content of *DO.Motor.p0180* is read out and the transferred DDS number is checked for validity
- The current content of *DO.Motor.p0573* is read out
- The MDS and EDS1 numbers belonging to the DDS number (*DO.Motor.p0186* and *DO.Motor.p0187*) are read out
- The motor commissioning parameter filter is set to **quick commissioning** (*Do.Antrieb.p0010 = 1*)
- *DO.Motor.p0573* is set to indicate whether or not the motor reference parameters (*Do.Motor.p2000 – Do.Motor1.p2006*) should be recalculated
- All motor parameters transferred via the input structure for the MDS and EDS1 identified from the DDS are written
- Completion of quick commissioning (*DO.Antrieb1.p3900 = 3*). Please see the documentation for parameter *p3900* for a description of what this involves. All the parameters of the associated motor are recalculated. If *disableCalculateReferenceParameter = TRUE*, the associated reference parameters are not recalculated. You can use the **FBLMoMaSetInhibitListForMotorCalculation** FB to exclude more motor parameters specifically from the recalculation
- Following recalculation, the old value for *DO.Motor.p0573* is written back
- The FB signals **ready** as soon as parameters *Do.Antrieb1.p0010* and *Do.Antrieb2.p0010* signal **ready to run** again following completion of quick commissioning

The encoder code number to be transferred to the FB can be determined in the configuration wizard of the associated drive.

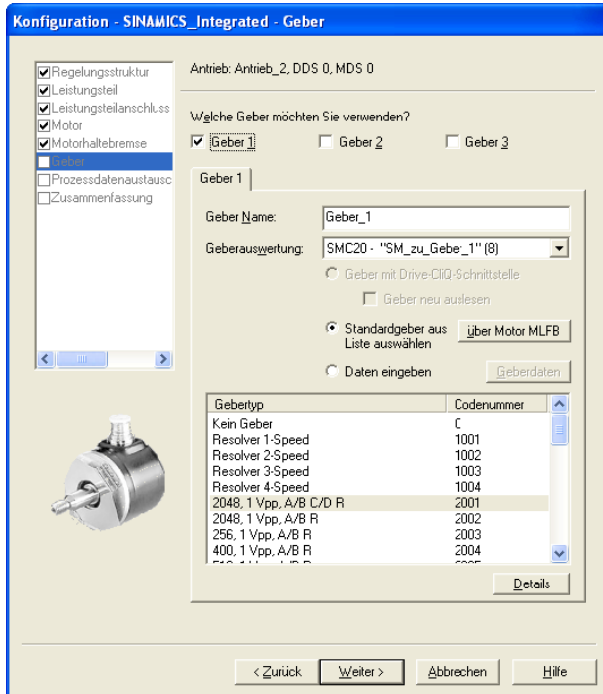


Figure 3-13 Encoder code numbers in the drive configuration wizard

Acyclic communication

Internally, the function block uses acyclic communication **DPV1** to write the parameters. For this reason, the FB must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for DPV1 services from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

3.4.8.2 Schematic diagram in LAD

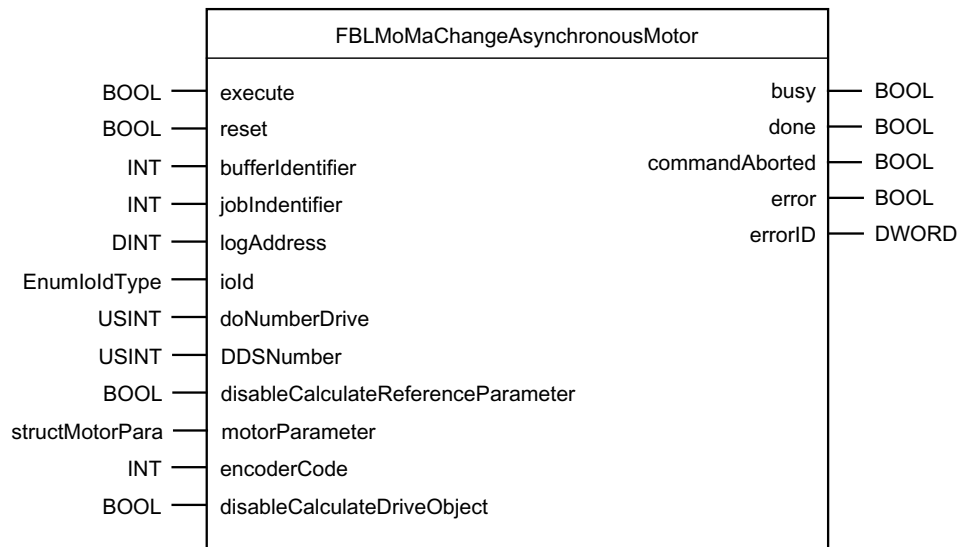


Figure 3-14 Schematic LAD diagram for FBLMoMaChangeAsynchronousMotor

3.4.8.3 Input and output parameters

Table 3- 16 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start status check/setting
reset	IN	BOOL	O	FALSE	Cancel function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of the SINAMICS drive on which the motor is to be changed. Alternatively, the diagnostics address can be specified.
iold	IN	EnumIold type	O	INPUT	Specification of the data direction of the logical address (input/output)
doNumberDrive	IN	USINT	O	255	Drive object number of the drive that is to be changed. If the drive object is transferred via the base address of the drive in <i>logAddress</i> , <i>doNumber</i> does not have to be supplied.
DDSNumber	IN	UINT	M	0	Data set number for whose MDS and EDS1 the motor is to be changed

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
disableCalculateReferenceParameter	IN	BOOL	O	TRUE	Prevents the recalculation of the reference parameters from <i>p2000</i> through <i>p2006Do.LM.p0573</i>
motorParameter	IN	structMotorPara	M		Structure of all required motor parameters for the asynchronous motor
encoderCode	IN	INT	O	-1	Code of the encoder. If no code is transferred or the code is -1, no changes are made to the encoder.
disableCalculateDriveObject	IN	BOOL	O	FALSE	If the input is set to TRUE, recalculation of the drive object is not triggered
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Function or job cancelled by a source external to the FB. Not relevant.
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

- 1) Parameter types: IN = input parameter, OUT = output parameter
- 2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 17 Motor parameters of the *motorParameter* structure with the *StructStartupOutputData* data type

Parameter	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
r32RatedMotorVoltageP304	IN	REAL	M	0	Rated motor voltage (type plate)
r32RatedMotorCurrentP305	IN	REAL	M	0	Rated motor current (type plate)
r32RatedMotorPowerP307	IN	REAL	M	0	Rated motor power (type plate)
r32RatedMotorPowerFactorP308	IN	REAL	M	0	Rated motor power factor (type plate)
r32RatedMotorFrequencyP310	IN	REAL	M	0	Rated motor frequency (type plate)
r32RatedMotorSpeedP311	IN	REAL	M	0	Rated motor speed (type plate)
r32MaximumMotorSpeedP322	IN	REAL	M	0	Maximum speed
i16MotorCoolingTypeP335	IN	INT	M	0	Motor cooling type
i16TechnologyApplicationP500	IN	INT	M	0	Technology application

- 1) Parameter types: IN = input parameter
- 2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.9 FBLMoMaSetInhibitListForMotorCalculation function block

3.4.9.1 General information

Functionality

The **FBLMoMaSetInhibitListForMotorCalculation** function block is used to set and activate/deactivate a blocked list for specific motor parameters. When the blocked list is activated, the values entered in it are not recalculated when the motor is recalculated. They retain the value that was valid prior to recalculation. Activating the blocked list and setting all possible blocking parameters to FALSE at the same time activates the blocked list currently set in the corresponding drive object. Please see the documentation for the *DO.Motor.p0571* for a list of the parameters of a motor that can be excluded from a recalculation by setting the blocked list. A blocked list can only be set for motor data set 0 (index 0).

Basic sequence of the FB

The FB has the following basic sequence:

- The values transferred in the input structure with TRUE are entered in the *DO.Motor.p0571* list in the addressed motor.
- In addition, the number of parameters set in *DO.Motor.p0570* is written.
- The set blocked list for the *DriveDataSet* transferred in *DDSNumber* is activated or deactivated in *DOx.p0572[DDSNumber]* dependent upon the FB call.
- The FB signals **ready** as soon as the actions have been completed.

Acyclic communication

Internally, the function block uses acyclic communication **DPV1** to write the parameters. For this reason, the FB must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for DPV1 services from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

3.4.9.2 Schematic diagram in LAD

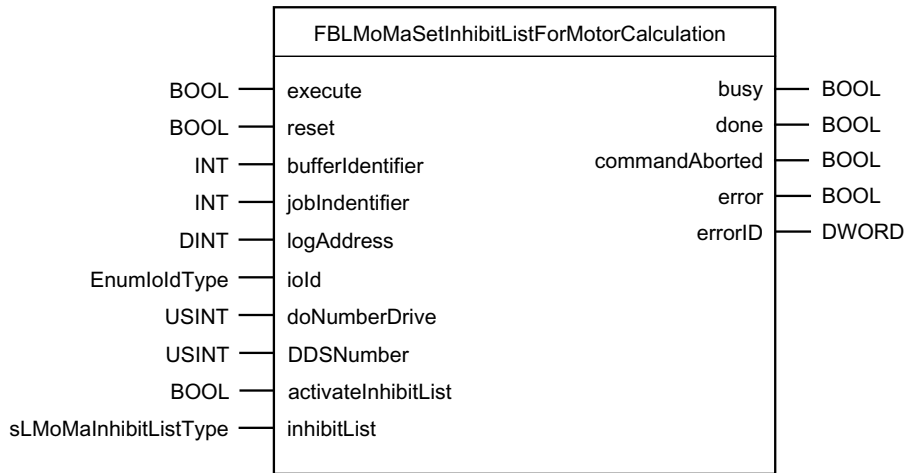


Figure 3-15 Schematic LAD representation

3.4.9.3 Input and output parameters

Table 3- 18 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start status check/setting
reset	IN	BOOL	O	FALSE	Cancel function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical SINAMICS address on which the blocked list is to be set. Alternatively, the diagnostics address can be specified.
iold	IN	EnumIold type	O	input	Specification of the data direction of the logical address (input/output)
doNumberDrive	IN	USINT	O	255	Drive object number of the motor to which the motor module is assigned. If the drive object is transferred via the base address of the drive in <i>logAddress</i> , <i>doNumber</i> does not have to be supplied.
DDSNumber	IN	USINT	M	0	Data set number for whose MDS and EDS1 the motor is to be changed
activateInhibitList	IN	BOOL	O	FALSE	Setting <i>activateInhibitList = TRUE</i> activates the blocked list. (The parameters selected in <i>inhibitList</i> are not recalculated when the motor is recalculated; they retain their old values.) Setting <i>activateInhibitList = FALSE</i> deactivates the blocked list.
inhibitList	IN	sLMoMain inhibitListType	M		Transfer structure with all settable parameters for the blocked list. Setting to TRUE sets the corresponding parameter. Setting to FALSE means that the corresponding parameter is not set.
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Function or job cancelled by a source external to the FB. Not relevant.
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 19 Input structure of the *sLMoMaInhibitListType* structure

Parameter	P type ¹⁾	Data type	Initial value	Meaning
boSpeedAtStartP348	IN	BOOL	FALSE	Speed at the start of field weakening 600 VDC
boCurrentLimitP640	IN	BOOL	FALSE	Current limit
boMaximumSpeedP1082	IN	BOOL	FALSE	Maximum speed
boSpeedSmoothingTimeP1441	IN	BOOL	FALSE	Speed actual value smoothing time
boSpeedControllerPGainP1460	IN	BOOL	FALSE	Speed controller P gain
boSpeedControllerIActionTimeP1462	IN	BOOL	FALSE	Speed controller integral time
boSpeedControllerPGainSensorlessP1470	IN	BOOL	FALSE	Speed controller P gain without encoder
boSpeedControllerIActionTimeSensorlessP1472	IN	BOOL	FALSE	Speed controller integral time without encoder
boTorqueLimitUpperP1520	IN	BOOL	FALSE	Upper torque limit/motor mode
boTorqueLimitLowerP1521	IN	BOOL	FALSE	Lower torque limit/generator mode
boPowerLimitMotoringP1530	IN	BOOL	FALSE	Power limit motor mode
boPowerLimitRegenerationP1531	IN	BOOL	FALSE	Power limit generator mode
boFluxControllerPGainP1590	IN	BOOL	FALSE	Flux controller P gain
boFluxControllerIActionTimeP1592	IN	BOOL	FALSE	Flux controller integral time
boActivateCurrentSetPointFilterP1656	IN	BOOL	FALSE	Current setpoint filter activation
boSpeedThreshold1P2141	IN	BOOL	FALSE	Speed threshold value 1
boHysteresisSpeed1P2142	IN	BOOL	FALSE	Hysteresis speed 1

1) Parameter types: IN = input parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.10 FBLMoMaSetBrakeConfiguration function block

3.4.10.1 General information

Functionality

The **FBLMoMaSetBrakeConfiguration** function block enables the user to change the parameterization of the motor holding brake for the drive system during runtime. Only the parameters for basic brake control are affected. Advanced brake control and SBC (Safe Brake Control) cannot be manipulated.

Basic sequence of the FB

The FB has the following basic sequence:

- All motor brake parameters transferred via the input structure are written
- The FB is terminated once the write operation is complete

Acyclic communication

Internally, the function block uses acyclic communication **DPV1** to write the parameters. For this reason, the FB must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for **DPV1 services** from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

Note

If an external motor holding brake is used, $p1215 = 3$ must be set and $r0899.12$ interconnected as the control signal.

If $p2038$ (PROFIdrive control word/status word interface mode/PD control word/status word IF mode) = 0. The state of the brake is transferred to SINAMICS in status word 1. The external motor holding brake can then be controlled in the higher-level control.

A BICO interconnection of the control signal is not implemented in the FB during runtime.

3.4.10.2 Schematic diagram in LAD

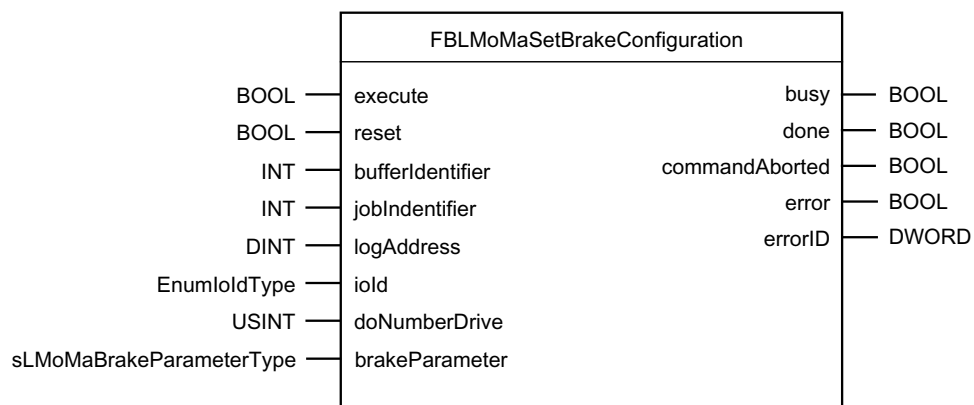


Figure 3-16 Schematic LAD diagram for FBLMoMaSetBrakeConfiguration

3.4.10.3 Input and output parameters

Table 3- 20 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start status check/setting
reset	IN	BOOL	O	FALSE	Cancel function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of a SINAMICS drive object on which the motor holding brake is to be reconfigured. Alternatively, the diagnostics address can be specified.
iold	IN	EnumIold type	O	input	Specification of the data direction of the logical address (input/output)
doNumberDrive	IN	USINT	O	255	Drive object number of the motor to which the motor module is assigned. If the drive object is transferred via the base address of the drive in <i>logAddress</i> , <i>doNumber</i> does not have to be supplied.
brakeParameter	IN	sLMoMaBrakeParameterType	M	-	Structure of all required brake parameters
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Function or job cancelled by a source external to the FB. Not relevant.
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

¹⁾ Parameter types: IN = input parameter, OUT = output parameter

²⁾ Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 21 Input structure of the *sLMoMaBrakeParameterType* structure

Parameter	P type ¹⁾	Data type	M/O type ²⁾	Initial value	Meaning
i16MotorHoldingBrakeConfigurationP1215	IN	INT	M	0	Motor holding brake configuration
r32MotorHoldingBrakeOpening timeP1216	IN	REAL	M	0	Motor holding brake opening time
r32MotorHoldingBrakeClosing timeP1217	IN	REAL	M	0	Motor holding brake closing time
r32ThresholdForZeroSpeedDetectionP1226	IN	REAL	M	0	Threshold for zero speed detection
r32ZeroSpeedDetectionMonitoringTimeP1227	IN	REAL	M	0	Zero speed detection monitoring time
r32PulseCancelationDelayTimeP1228	IN	REAL	M	0	Pulse suppression delay time
i16BrakeControlDiagnosticsEvaluationP1278	IN	INT	M	0	Brake control diagnostics evaluation

1) Parameter types: IN = input parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.11 FBLMoMaResetAndLoadParameterFromStorageMedium function block

3.4.11.1 General information

Functionality

The **FBLMoMaResetAndLoadParameterFromStorageMedium** function block enables the user to reset a SINAMICS component during runtime and then reload all of the drive system parameters from the non-volatile memory (ROM) on restarting.

Basic sequence of the FB

The FB has the following basic sequence:

- The component reset is prepared by writing *CU.p0009 = 30*
- The reset is started by writing *CU.p0976 = 2*. The parameters stored in the non-volatile memory start to be loaded
- Parameters *CU.p0009*, *CU.p0976*, and *CU.r0002* are read out. Once these parameters can be read again and have the values *p0976 = 0*, *p0009 = 0*, and *r0002 <> 20* (waiting for ramp-up), the FB signals **ready**

Acyclic communication

Internally, the function block uses acyclic communication **DPV1** to write the parameters. For this reason, the FB must be called from within a cyclic task. To ensure reliable use of acyclic communication, we recommend using buffer management for DPV1 services from the **LDPV1** library. If this is not used, the user is responsible for ensuring that only one instance of acyclic communication per device is running at any one time.

3.4.11.2 Schematic diagram in LAD

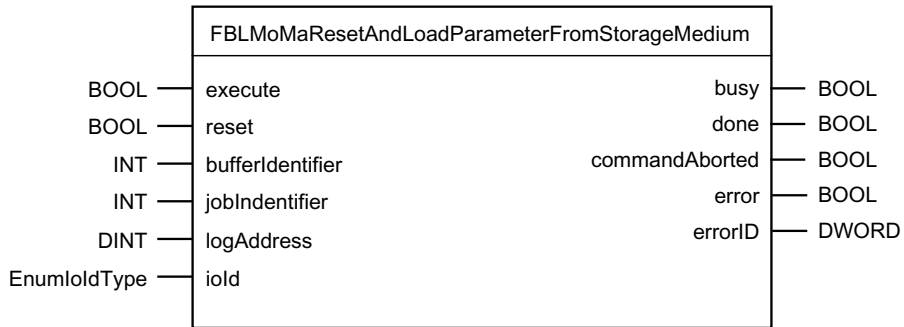


Figure 3-17 Schematic LAD diagram for FBLMoMaResetAndLoadParameterFromStorageMedium

3.4.11.3 Input and output parameters

Table 3- 22 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start status check/setting
reset	IN	BOOL	O	FALSE	Cancel function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of a SINAMICS drive object on which the restart is to be executed. Alternatively, the diagnostics address can be specified.
iold	IN	EnumIold type	O	INPUT	Specification of the data direction of the logical address (input/output)
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.12 FBLMoMaCopySIMOTIONConfigDataToROM function block

3.4.12.1 General information

Functionality

The **FBLMoMaCopySIMOTIONConfigDataToROM** function block saves changes to the configuration data of TOs to the RAM disk **Copy current data to RAM**) and then copies the RAM disk to the ROM. Copy RAM to ROM only affects SIMOTION data.

NOTICE

The function block can only be started in a motion task. The runtime of the FB would otherwise cause errors in runtime monitoring.

3.4.12.2 Schematic diagram in LAD

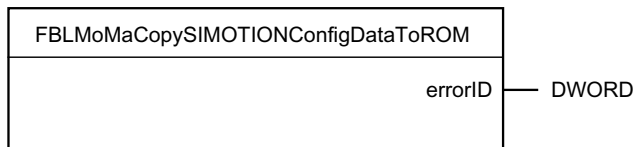


Figure 3-18 Schematic LAD diagram for FBLMoMaCopySIMOTIONConfigDataToROM

3.4.12.3 Input and output parameters

Table 3-23 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
errorID	OUT	DWORD	-	0	The ID can be used to specify an error that has occurred

¹⁾ Parameter types: OUT = output parameter

²⁾ Parameter category: M = mandatory parameter, O = optional parameter

3.4.13 FBLMoMaSetActiveDDS function block

3.4.13.1 General information

Functionality

The **FBLMoMaSetActiveDDS** function block enables the active data set of a drive to be changed over via the application software. When an axis instance is transferred, the FB takes over the handling of the `_setAxisSTW` system function for the user. `_setAxisSTW` can be used to set the data set to be used by the drive object with drive cyclic communication. However, this can only be done if a standard message frame is used in the configuration. If a data set is to be switched over at a drive object without cyclic communication or with its own message frame configuration (the use of `_setAxisSTW` is not permitted (possible) in this case), the corresponding drive object must be addressed directly via its logical base address or the DO number. In this case, the parameters of the drive object are accessed directly with acyclic communication. In the case of user-defined message frame configuration, however, no user-defined wiring settings can be made for parameters `DOx.p0820 – DOx.p0824`, as these are overwritten.

When an axis instance is transferred, the function block first reads `DOx.0180` to determine whether the DDS to be activated is actually available. At the same time, the set interface mode is determined by reading `DOx.p2038`. The interface mode can then be used to distinguish which bits must be set within cyclic communication to transfer the data set selected with `DDSNumber` correctly.

The following modes are used:

- **SIMODRIVE 611universal**
This mode is set automatically by the system and cannot be changed when selecting message frame 105, for example. DDS selection is made using *bits 0 - 2* and *bits 9 - 10* of *control word 2*
- **SINAMICS**
This mode is set by the system when message frame 2 is selected, for example. DDS selection is made using *bits 0 - 4* of *control word 2*
- **VIK-Namur**
DDS switchover is not possible
- Processing is completed by reading out parameter `DOx.r0051`. This parameter is used to check whether the previously set DDS is actually effective.
- In the case of drive objects without cyclic communication or with user-defined message frame configuration, the DDS number to be activated is written directly to `DOx.p0820 – DOx.p0824`. However, this can only be done if no user-defined wiring settings have been made for parameters `DOx.p0820 – DOx.p0824`, as these are overwritten

3.4.13.2 Schematic diagram in LAD

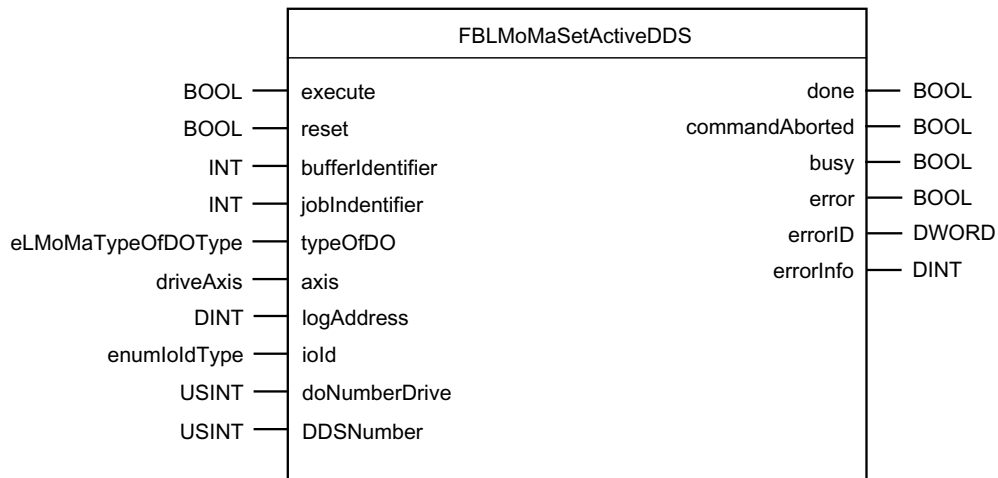


Figure 3-19 Schematic LAD diagram for FBLMoMaSetActiveDDS

3.4.13.3 Input and output parameters

Table 3- 24 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop the function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
typeOfDo	IN	eLMoMaTypeOfDOType	O	DO_WITH_TO	Indicates the drive object type on which the active DDS is to be switched over. The following values are possible: DO_WITH_STANDARD_TELEGRAM DO_WITHOUT_STANDARD_TELEGRAM
axis	IN	driveAxis	O	TO#NIL	Transfers the axis TO at which data set switchover is to be performed. Different scenarios dependent upon <i>typeOfDO</i> 1: DO_WITH_STANDARD_TELEGRAM Transfer of axis instance 2: DO_WITHOUT_STANDARD_TELEGRAM No significance

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
logAddress	IN	DINT	M	0	Logical address of the drive object, different scenarios dependent upon <i>typeOfDO</i> 1: DO_WITH_STANDARD_TELEGRAM No significance 2: DO_WITHOUT_STANDARD_TELEGRAM Any logical address of the device on which the drive object is located. If no DO number can be transferred, the logical base address must be transferred
iold	IN	EnumIold type	O	INPUT	Specification of the data direction of the base address (input/output) 1: DO_WITH_STANDARD_TELEGRAM No significance 2: DO_WITHOUT_STANDARD_TELEGRAM
doNumberDrive	IN	USINT	O	255	DO number of the drive object on which the DDS is to be set. 2 different scenarios: For 1: DO_WITH_STANDARD_TELEGRAM Not relevant 2: DO_WITHOUT_STANDARD_TELEGRAM Drive object number of the drive object without control word 2. If the DO is transferred via the base address of the drive in <i>logAddress</i> , <i>doNumberDrive</i> does not have to be supplied or can be set to a value of 255.
DDSNumber	IN	USINT	O	0	Data set number to be activated
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Processing cancelled by a source external to the FB
busy	OUT	BOOL	-	FALSE	Function block being processed
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code
errorInfo	OUT	DINT	-	0	Additional error information during internal processing of system functions

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 25 Enum for the *eLMoMaTypeOfDOType* structure

Enum identifier	Comment
DO_WITH_STANDARD_TELEGRAM	Drive objects with technology object, coupled via standard message frame (only drives/axes in this case)
DO_WITHOUT_STANDARD_TELEGRAM	Drive objects without standard message frame

Note

The data type of the *axis* input variable has been dimensioned with *driveAxis* here so that only axis type TO instances can be transferred. The *driveAxis* data type is the basis of all other axis types that can be created in SIMOTION. Therefore, *posAxis* and *followingAxis* type axes can always be transferred to a *driveAxis* type variable without generating a type conflict in the compiler.

3.4.14 FBLMoMaSetResetParkingAxis function block

3.4.14.1 General information

Functionality

The **FBLMoMaSetResetParkingAxis** function block is used to switch a configured motor to **parking axis** operating mode.

The *parking axis* state is defined as follows:

- Monitoring of all encoders and motor modules assigned to the **Motor control** application of a drive is suppressed
- All encoders assigned to the **Motor control** application of a drive are set to the **Encoder removed** state
- The motor module assigned to the **Motor control** application of a drive is prepared for the **Motor module removed** state.
- When an axis is parked, the power unit and all encoders assigned to **Motor control** are switched to inactive.

The basic sequence of the software for drive objects with technology objects using a standard message frame is as follows:

- **Activate parking axis:**
 - The transferred axis is set to **parking axis** using the system command `_setAxisSTW`. The parking axis is set/reset in *control word 2.7*. SINAMICS sets the associated drive object to inactive when **parking axis** is selected in `DOx.r0126` and `DOx.r0146`. The FB terminates processing once it has unequivocally read back the **parking axis** state of the drive object via bit `DOx.r0896.0`
 - Since **parking axis** is set via cyclic communication between TO and drive object, the **parking axis active** state cannot be saved to non-retentive memory and must, therefore, be set by the application again every time the machine is restarted.
- **Deactivate parking axis:**
 - If the **parking axis** state has been set and the associated drive is not plugged in, the system outputs warnings at the drive. These signal that the drive is not ready to run and that components may be missing from the actual topology. To achieve a reproducible response when deactivating **parking axis**, the FB first determines whether the drive and all its components are actually present. It does this by reading the alarm buffer and checking that warnings `A01481` (power unit missing) and `A01482` (sensor module missing) are not pending. The **unparking** of the motor is delayed until the motor is reported as **present**. If the drive is displayed as **present**, the **parking axis** state is canceled with `_setAxisSTW`. The FB terminates processing once it has read back via bit `DOx.r0896.0` that the **parking axis** state is no longer active. In addition, a check is made to ascertain whether an encoder interface 1 has been configured at the currently active DDS and whether it is set to **activate**. If this is the case, the FB waits until the encoder interface and the power unit return their states as **active** in `r0126` and `r0146`. Only then does the FB signal the end of processing and the drive is ready for traversing. If an encoder interface 1 has not been configured or if the configured encoder interface is set to **inactive and not present**, the active state of the power unit is also checked.

The checkback signal indicating the status of the axis could be sent with cyclic communication (status word 2.7). However, so that the software sequence is uniform for all types of drive object, the parameters of the drive object are accessed directly by means of acyclic communication.

- If a drive object without cyclic communication or with user-defined message frame configuration is to be set to **parking** (`_setAxisSTW` cannot be used in this case), the corresponding drive object must be addressed directly via its logical base address or the DO number. In this case, the parameters of the DO are accessed directly with acyclic communication. However, in this case, no user-defined wiring settings may be made at parameter `DOx.p0897`, as these are overwritten.
- In the case of drive objects without cyclic communication or with user-defined message frame configuration, the required state is written directly to `DOx.p0897`. The checkback signal indicating that processing has been completed successfully is also implemented by reading `DOx.r0896`.

Note

If a drive object is set to the **parking axis** state via cyclic communication, the associated TO must not be deactivated. Deactivating the associated TO would stop the **parking axis active** state set via cyclic communication from being transferred to the drive object and, therefore, reset the drive object to the active state.

3.4.14.2 Schematic diagram in LAD

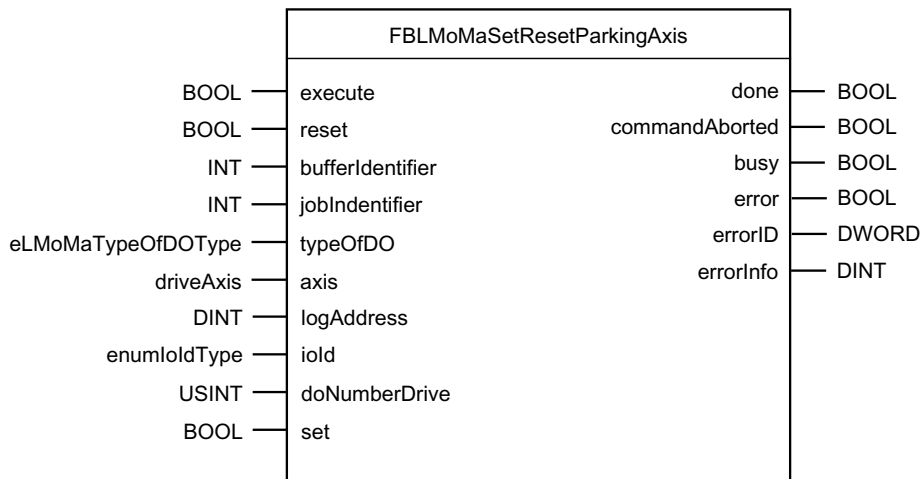


Figure 3-20 Schematic LAD diagram for FBLMoMaSetResetParkingAxis

3.4.14.3 Input and output parameters

Table 3- 26 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop the function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
typeOfDo	IN	eLMoMaT typeOfDOT ype	O	DO_WITH_TO	Indicates the drive object type on which the active DDS is to be switched over. The following values are possible: DO_WITH_STANDARD_TELEGRAM DO_WITHOUT_STANDARD_TELEGRAM
axis	IN	driveAxis	O	TO#NIL	Transfers the axis TO at which data set switchover is to be performed. Different scenarios dependent upon <i>typeOfDO</i> 1: DO_WITH_STANDARD_TELEGRAM Transfer of axis instance 2: DO_WITHOUT_STANDARD_TELEGRAM No significance
logAddress	IN	DINT	M	0	Logical address of the drive object, different scenarios dependent upon <i>typeOfDO</i> 1: DO_WITH_STANDARD_TELEGRAM No significance 2: DO_WITHOUT_STANDARD_TELEGRAM Any logical address of the device on which the drive object is located. If no DO number can be transferred, the logical base address must be transferred
iold	IN	EnumIold type	O	INPUT	Specification of the data direction of the base address (input/output) 1: DO_WITH_STANDARD_TELEGRAM No significance 2: DO_WITHOUT_STANDARD_TELEGRAM
doNumberDrive	IN	USINT	O	255	DO number of the drive object on which the DDS is to be set. 2 different scenarios: For 1: DO_WITH_STANDARD_TELEGRAM Not relevant 2: DO_WITHOUT_STANDARD_TELEGRAM Drive object number of the drive object without control word 2. If the drive object is transferred via the base address of the drive in <i>logAddress</i> , <i>doNumberDrive</i> does not have to be supplied or can be set to a value of 255.

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Processing cancelled by a source external to the FB
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code
errorInfo	OUT	DINT	-	0	Additional error information during internal processing of system functions
set	IN	BOOL	O	FALSE	If <i>set</i> = <i>TRUE</i> , the addressed drive is set to parking axis . If <i>set</i> = <i>FALSE</i> , parking axis is reset.

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 27 Enum for the *eLMoMaTypeOfDOType* structure

Enum identifier	Comment
DO_WITH_STANDARD_TELEGRAM	Drive objects with technology object, coupled via standard message frame (only drives/axes in this case)
DO_WITHOUT_STANDARD_TELEGRAM	Drive objects without standard message frame

Note

The data type of the *axis* input variable has been dimensioned with *driveAxis* here so that only axis type TO instances can be transferred. The *driveAxis* data type is the basis of all other axis types that can be created in SIMOTION. Therefore, *posAxis* and *followingAxis* type axes can always be transferred to a *driveAxis* type variable without generating a type conflict in the compiler.

3.4.15 FBLMoMaActivateDeactivateSingleDOComponents function block

3.4.15.1 General information

Functionality

The **FBLMoMaActivateDeactivateSingleDOComponents** function block is used to activate/deactivate individual components of a drive object. This can be the case, for example, if the power unit is present for a drive object but the associated motor is not. For a drive with an encoder, the encoder interface would then be missing and would have to be set to inactive and not present separately.

The function block sets the state transferred to *condition* at the corresponding drive object on the basis of the component specified in *componentType*.

A distinction is made between the following components:

- Power unit for A_INF, B_INF, SERVO, S_INF, and VECTOR (DOx.p0125/DOx.r0126)
- Encoder interface for SERVO, VECTOR (DOx.p0145/DOx.r0146)
- Voltage sensing module for A_INF, S_INF (DOx.p0145/DOx.r0146)
- Voltage sensing module for VECTOR (DOx.p0155/DOx.r0156)
- Entire drive object (DOx.p0105/DOx.r0106)

The function block sets the required state of the component and terminates processing if the component adopts the required state.

If components of other configured *DriveDataSets* are to be processed, the corresponding data set number must be transferred to the FB. In this case the FB obtains the corresponding subindex of the component and activates or deactivates it.

Example:

MDS0, EDS0, and EDS1 have been configured in DDS0. MDS1, EDS0, and EDS3 have been configured in DDS1. To deactivate the encoder interface of EDS3 at this point, input *componentType = ENCODER_INTERFACE_3* and *DDSNumber = 1* must be set. The *A_INF*FEED and *VECTOR* (motor module) devices support parallel connections. If a specific device in the parallel connection is to be processed, the FB must be informed accordingly via the *PDSNumber* input. Entering this information also generates the subindex of the component to be processed. In the case of DDS or PDS transfer, a check is first made to ascertain whether or not the transferred data set is present. If it is, the FB commences processing.

3.4.15.2 Schematic diagram in LAD

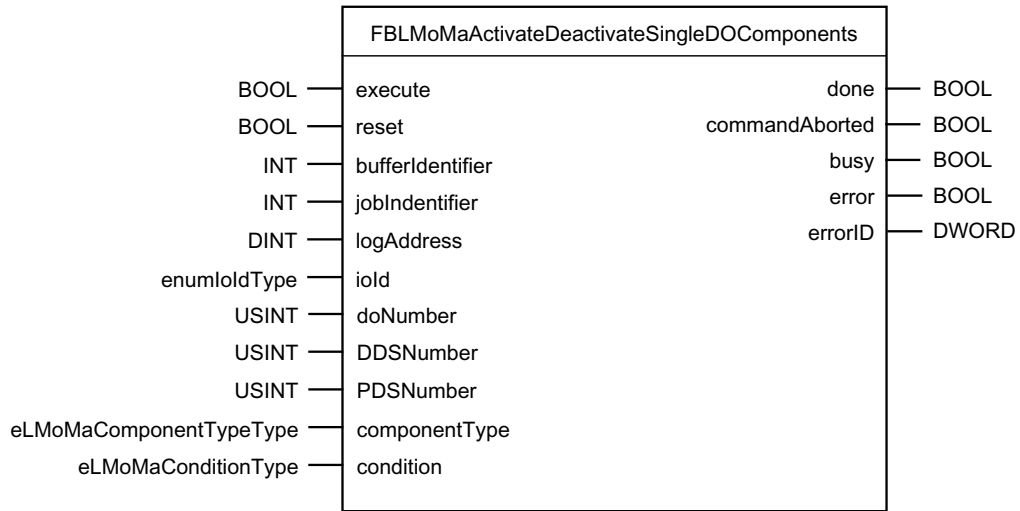


Figure 3-21 Schematic LAD diagram for FBLMoMaActivateDeactivateSingleDOComponents

3.4.15.3 Input and output parameters

Table 3- 28 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop the function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of the SINAMICS component on which the drive object is located. Alternatively, the diagnostics address can be specified. <i>doNumber</i> must be transferred with this type of addressing. A drive can also be specified via its base address; in this case <i>doNumber</i> does not have to be transferred, or 255.
iold	IN	EnumIold type	O	INPUT	Specification of the data direction of the logical address (input/output)
doNumber	IN	USINT	O	255	DO number for access to the SINAMICS component: <> 0: DO number = 0: Not permitted = 255 DO is only addressed via the logical address

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
DDSNumber	IN	USINT	O	0	Transfer of the <i>DriveDataSet</i> number for whose <i>MotorDataSet</i> number and <i>EncoderDataSet</i> number the corresponding component is to be processed
PDSNumber	IN	USINT	O	0	Transfer of the <i>PowerDataSet</i> number whose corresponding component is to be processed. Only relevant for <i>A_INF</i> and <i>vector</i> . For <i>vector</i> , the number of the <i>voltage sensing module</i> is also transferred here.
componentType	IN	eLMoMaComponentTypeType	O	WHOLE_DRIVE_OBJECT	Transfer of the component of the drive object which is to be activated/deactivated
condition	IN	eLMoMaConditionType	O	ACTIVATE	The intended purpose of the component is transferred here
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Processing cancelled by a source external to the FB
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code
errorInfo	OUT	DINT	-	0	Additional error information during internal processing of system functions

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 29 Enum for the *eLMoMaComponentTypeType* structure

Enum identifier	Value	Comment
WHOLE_DRIVE_OBJECT	0	Entire drive object via p0105
POWER_UNIT_COMPONENT	1	Power unit via p0125
ENCODER_INTERFACE_1	2	Encoder interface via p0145 for encoder 1
ENCODER_INTERFACE_2	3	Encoder interface via p0145 for encoder 2
ENCODER_INTERFACE_2	4	Encoder interface via p0145 for encoder 3
VOLTAGE_SENSING_MODUL	5	Voltage sensing module infeed via p145
VOLTAGE_SENSING_MODUL_VECTOR	6	Voltage sensing module drive vector p155

Table 3- 30 Enum for the *eLMoMaConditionType* structure

Enum identifier	Value	Comment
ACTIVATE	0	Activate component
DEACTIVATE	1	Deactivate component
DEACTIVATE_AND_NOT_PRESENT	2	Deactivate component and not present

3.4.16 FBLMoMaActivateDeactivateEncoder1AndSetActiveDDS function block

3.4.16.1 General information

Functionality

The **FBLMoMaActivateDeactivateEncoder1AndSetActiveDDS** function block is used to set the interface configured for encoder 1 to inactive and not present at the drive object belonging to the axis TO to be transferred. This setting is made using application software. Moreover, the data set transferred to *DDSNumber* via cyclic communication between TO and drive object is activated.

3.4.16.2 Schematic diagram in LAD

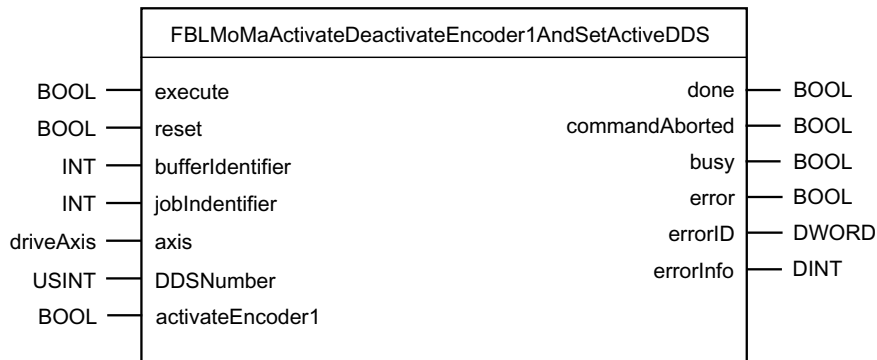


Figure 3-22 Schematic LAD diagram for FBLMoMaActivateDeactivateEncoder1AndSetActiveDDS

3.4.16.3 Input and output parameters

Table 3- 31 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop the function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
axis	IN	driveAxis	O	TO#NIL	Transfers the axis TO at which data set switchover is to be performed. Different scenarios dependent upon <i>typeOfDO</i> 1: DO_WITH_STANDARD_TELEGRAM Transfer of the axis instance 2: DO_WITHOUT_STANDARD_TELEGRAM No significance
DDSNumber	IN	USINT	O	0	Data set number to be activated
activateEncoder1	IN	BOOL	O	FALSE	FALSE sets the <i>EncoderInterface</i> to inactive and not present. TRUE sets the <i>EncoderInterface</i> to active.
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Processing cancelled by a source external to the FB
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code
errorInfo	OUT	DINT	-	0	Additional error information during internal processing of system functions

1) Parameter types: IN = input parameter, OUT = output parameter, IN/OUT = in/out parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Note

The data type of the *axis* input variable has been dimensioned with *driveAxis* here so that only axis type TO instances can be transferred. The *driveAxis* data type is the basis of all other axis types that can be created in SIMOTION. Therefore, *posAxis* and *followingAxis* type axes can always be transferred to a *driveAxis* type variable without generating a type conflict in the compiler.

3.4.17 FBLMoMaChangeTOConfigData function block

3.4.17.1 General information

Functionality

The **FBLMoMaChangeTOConfigData** function block is used to write the following configuration data to the transferred axis TO.

Table 3- 32 Configuration data written

Configuration data element	Designation
TypeOfAxis.SetPointDriverInfo.DriveData.nominalspeed	Rated speed
TypeOfAxis.SetPointDriverInfo.DriveData.maxspeed	Max. speed
TypeOfAxis.SetPointDriverInfo.DriveData.maxTorque	Rated torque
TypeOfAxis.NumberOfDataSets.DataSet_1.Gear.NumFactor	Gear ratio motor revolutions
TypeOfAxis.NumberOfDataSets.DataSet_1.Gear.DenFactor	Gear ratio load revolutions
LeadScrew.PitchVal	Leadscrew pitch per revolution of the axis for linear axes
TypeOfAxis.SetPointDriverInfo.lifeSignCheck	Life-sign monitoring drive
TypeOfAxis.NumberOfEncoders.Encoder_1.DriverInfo.lifeSignCheck	Life-sign monitoring encoder

Moreover, the value for *TypeOfAxis.maxvelocity.maximum* (maximum velocity) is recalculated and also written to the TO.

The calculation is made according to the following formulas, which are dependent upon when the axis is linear or rotary.

For a rotary axis:

$$\text{TypeOfAxis.maxvelocity.maximum}[\text{°/s}] = \text{TypeOfAxis.SetpointDriverInfo.DriveData.maxSpeed} [\text{rpm}] * \text{gear ratio} * 1/60 * 360^\circ$$

For a linear axis:

$$\text{TypeOfAxis.maxvelocity.maximum}[\text{mm/s}] = \text{TypeOfAxis.SetpointDriverInfo.DriveData.maxSpeed} [\text{rpm}] * \text{gear ratio} * 1/60 * \text{leadscrew pitch per revolution} [\text{mm}]$$

For a speed-controlled axis:

$$\text{TypeOfAxis.maxvelocity.maximum}[\text{mm/s}] = \text{TypeOfAxis.SetpointDriverInfo.DriveData.maxSpeed} [\text{rpm}] * \text{gear ratio}$$

The configuration data valid after a TO restart is always used to make the calculation.

Before writing the configuration data, the function block checks that **Normalize to maximum motor speed** has not been selected. Processing of the FB does not continue if this condition has not been met.

Basic sequence of the FB

The subsequent configuration data write operation proceeds as follows:

- The rated speed, maximum speed, and rated torque are written
- If the Boolean value of the *configData* input structure belonging to the corresponding value is set to TRUE (*boWriteNominalSpeed*, *boWriteMaxSpeed* or *boWriteMaxTorque*), the FB writes the value specified under this in the structure to the corresponding configuration variable. If the Boolean value in *configData* is set to FALSE, the FB writes the value belonging to the configuration variable configured in the drive object. For this purpose, the corresponding parameter of the drive objects is read out (*p0311*, *p0322*, *p0312*) and transferred to the configuration variable taking *DDSNumber* into account
- The gear ratio is written
- If *boWriteGearFactor* is set to TRUE in *configData*, both values belonging to the gear ratio are always written to the TO. If *boWriteGearFactor* = FALSE, the old gear ratio is retained
- *LifeSignCheck* is written
- Data is only written if the corresponding Boolean values in *configData* are set to true
- Finally, the FB executes a TO restart and terminates processing as soon as the TO restart has been completed

3.4.17.2 Schematic diagram in LAD

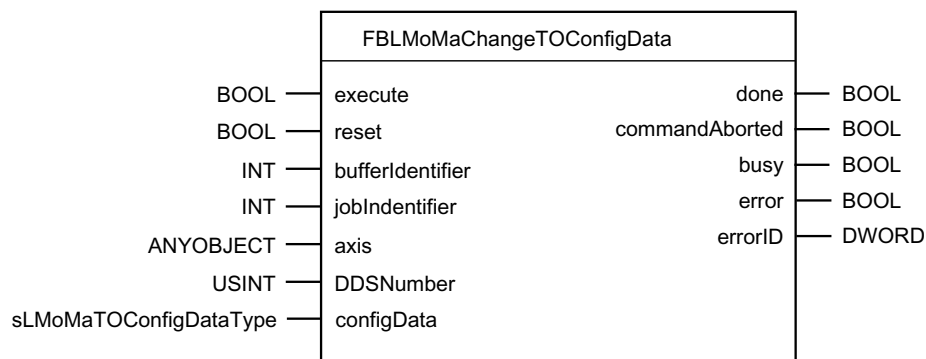


Figure 3-23 Schematic LAD diagram for FBLMoMaChangeTOConfigData

3.4.17.3 Input and output parameters

Table 3- 33 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop the function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
axis	IN	ANYOBJE CT	O	TO#NIL	Transfer of the axis TO at which the configuration data is to be changed
DDSNumber	IN	USINT	O	0	Data set number on the basis of which configuration data is to be changed
configData	IN	sLMoMaT OConfigD ataType	O	-	Structure for transferring new TO configuration data
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Processing cancelled by a source external to the FB
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 34 Input structure of the *sLMoMaTOConfigDataType* structure

Parameter	P type ¹⁾	Data type	M/O type ²⁾	Initial value	Meaning
boWriteNominalSpeed	IN	BOOL	O	FALSE	Write new rated speed?
r64NominalSpeedValue	IN	LREAL	O	0	New rated speed value in rpm
boWriteMaxSpeed	IN	BOOL	O	FALSE	Write new maximum speed?
r64MaxSpeedValue	IN	LREAL	O	0	New maximum speed value in rpm
boWriteMaxTorque	IN	BOOL	O	FALSE	Write new maximum torque?
r64MaxTorqueValue	IN	LREAL	O	0	New maximum torque value in Nm
boWriteGearFactor	IN	BOOL	O	FALSE	Write new gear ratio? If TRUE, the values for load and motor revolutions must be specified
u32GearDenFactorValue	IN	UDINT	O	0	New load revolutions value
u32GearNumFactorValue	IN	UDINT	O	0	New motor revolutions value
boWriteLeadSrewPitch	IN	BOOL	O	FALSE	Write new leadscrew pitch? Only possible for linear axes.
r64LeadSrewPitchValue	IN	LREAL	O	0	New leadscrew pitch value for linear axis
boWriteLifeSignCheckDrive	IN	BOOL	O	FALSE	Change life-sign monitoring at drive?
eValueDrive	IN	EnumYesNo	O	YES	New value for life-sign monitoring
boWriteLifeSignCheckEncoder	IN	BOOL	O	FALSE	Change life-sign monitoring at encoder?
eValueEncoder	IN	EnumYesNo	O	YES	New value for life-sign monitoring

1) Parameter types: IN = input parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.4.18 FBLMoMaActivateDeactivateTOAndEncoder1 function block

3.4.18.1 General information

Functionality

The **FBLMoMaActivateDeactivateTOAndEncoder1** function block makes the following possible by means of appropriate parameterization:

- Activate encoder interface 1 at a drive or set to inactive and not present
- Activate/deactivate a single axis TO with activation/deactivation of one or two following objects at the same time in the case of following axis type TOs. These objects are transferred to the FB via input parameters and activated/deactivated with the same block call. The speed-controlled axis TO, positioning axis TO, and following axis TO (including the following objects, of which there are up to two) can be deactivated.

This FB can be used to implement **hot plugging** of a motor. In this case, only the encoder interface including motor are removed from a machine and the associated motor module remains plugged in in the control cabinet.

TOs and encoder interface1 can be activated/deactivated independently or jointly.

The basic sequence of the software is as follows:

- **Activate: (*activate input = TRUE*)**
During activation, a check is performed first to determine whether the encoder interface is set to inactive or inactive and not present at the corresponding drive object. If it is, a check is then performed to see if warning *A01317* (deactivated component present again) is currently active. Only if this warning is pending can you be assured that the DRIVE-CLiQ cable of the SMI motor has actually been disconnected and plugged back in. After the warning has been detected, the encoder interface is switched back to active at the drive object. A check is then performed to ascertain whether *DOx.p0145* and *DOx.p0146* are displaying active. If they are, the associated TO and any following objects transferred are activated when an axis reference is transferred
- **Deactivate: (*activate input = FALSE*)**
During deactivation, a check is performed first to determine whether the encoder interface is set to inactive or active at the corresponding drive object. If it is, *DOx.p0145* is written to inactive and not present. Next, *DOx.p0145* and *DOx.p0145* are used to check whether the encoder interface (*DOx.p0146*) is inactive or (*DOx.p0145*) inactive and not present. If it is, the associated TO and any following objects transferred are deactivated when an axis reference is transferred. Finally, the warning buffer is read out at the CU belonging to the drive object until warning *A01314* (component must not be present) is no longer pending. Based on this warning, a check is made to ascertain whether the encoder interface has actually been unplugged and as such is no longer present

Note

The data type of the *axis* input variable has been dimensioned with *driveAxis* here so that only axis type TO instances can be transferred. The *driveAxis* data type is the basis of all other axis types that can be created in SIMOTION. Therefore, *posAxis* and *followingAxis* type axes can always be transferred to a *driveAxis* type variable without generating a type conflict in the compiler.

3.4.18.2 Schematic diagram in LAD

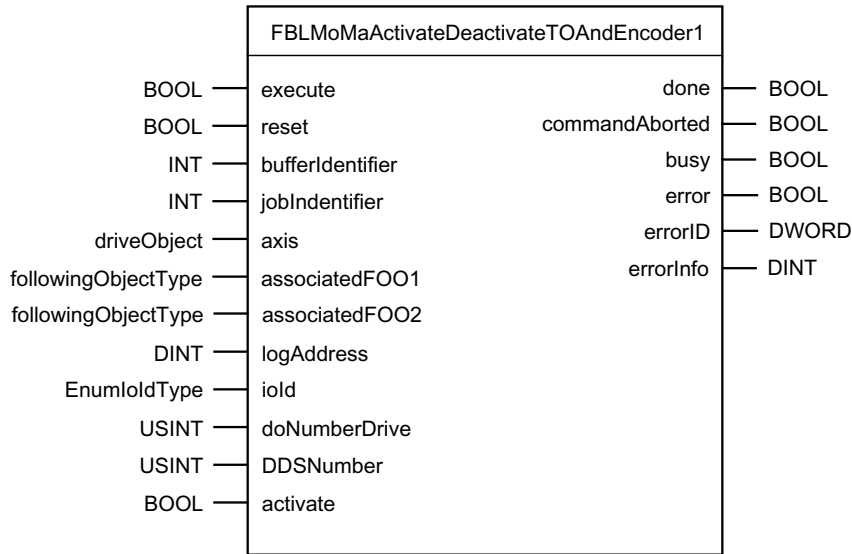


Figure 3-24 Schematic LAD diagram for FBLMoMaActivateDeactivateTOAndEncoder1

3.4.18.3 Input and output parameters

Table 3- 35 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop the function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
axis	IN	ANYOBJECT	O	TO#NIL	Transfer of the axis TO to be activated or deactivated. If TO#NIL is transferred, no action is taken at the TO
associatedFOO1	IN	Following Object	O	TO#NIL	In the case of following axes, the associated following object1 or TO#NIL
associatedFOO2	IN	Following Object	O	TO#NIL	In the case of following axes, the associated following object2 or TO#NIL
logAddress	IN	DINT	O	0	Any logical address of the device on which the drive object is located. If no DO number can be transferred, the logical base address must be transferred
iold	IN	enumIoldType	O	INPUT	Specification of the data direction of the transferred logical address

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
doNumberDrive	IN	USINT	O	255	DO number of the drive object without control word 2 If the DO is transferred via the base address of the drive in <i>logAddress</i> , <i>doNumberDrive</i> does not have to be supplied or can be set to a value of 255
DDSNumber	IN	USINT	O	0	Data set number on the basis of which configuration data is to be changed
activate	IN	BOOL	O	FALSE	If <i>activate = TRUE</i> , <i>Encoderinterface1</i> and the TO are activated in accordance with the parameterization. If <i>activate = FALSE</i> , <i>Encoderinterface</i> is set to inactive and not present and the TO is set to inactive.
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Processing cancelled by a source external to the FB
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Note

The data type of the *axis* input variable has been dimensioned with *driveAxis* here so that only axis type TO instances can be transferred. The *driveAxis* data type is the basis of all other axis types that can be created in SIMOTION. Therefore, *posAxis* and *followingAxis* type axes can always be transferred to a *driveAxis* type variable without generating a type conflict in the compiler.

3.4.19 FBLMoMaAcceptNewEncoderSerialNumber function block

3.4.19.1 General information

Functionality

The **FBLMoMaAcceptNewEncoderSerialNumber** function block is used to read in and apply the encoder serial number belonging to the transferred encoder data set.

If the serial numbers of other configured *DriveDataSets* are to be applied, the corresponding data set number must be transferred to the FB. In this case the FB obtains the corresponding subindex of the component and activates the reading in of the serial number.

Example:

MDS0, EDS0, and EDS1 have been configured in DDS0.

MDS1, EDS0, and EDS3 have been configured in DDS1.

To read in the encoder serial number of EDS3 at this point, input *encoderType = ENCODER_INTERFACE_3* and *DDSNumber = 1* must be set.

In the case of DDS transfer, a check is first made to ascertain whether or not the transferred data set is present. If it is, the FB commences processing.

3.4.19.2 Schematic diagram in LAD

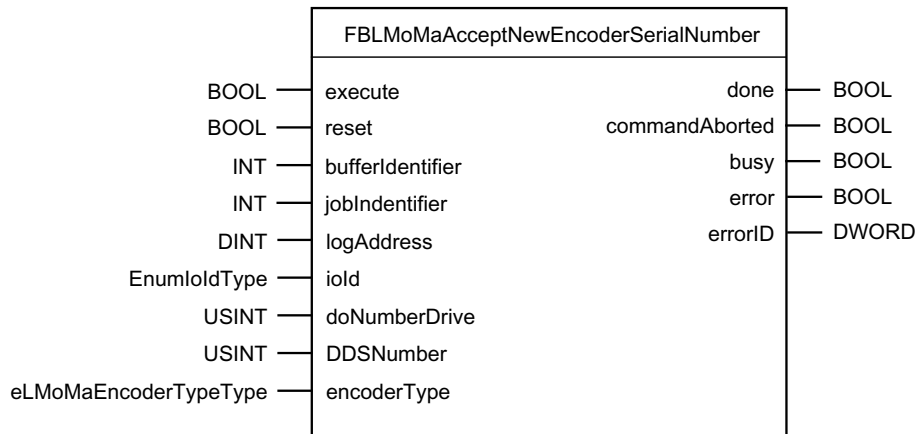


Figure 3-25 Schematic LAD diagram FBLMoMaAcceptNewEncoderSerialNumber

3.4.19.3 Input and output parameters

Table 3- 36 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop the function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	O	0	Any logical address of the SINAMICS component on which the drive object is located. Alternatively, the diagnostics address can be specified. <i>doNumber</i> must be transferred with this type of addressing. A drive can also be specified via its base address; in this case <i>doNumber</i> does not have to be transferred, or 255.
iold	IN	enumIoldType	O	INPUT	Specification of the data direction of the transferred logical address (input/output)
doNumberDrive	IN	USINT	O	255	DO number for access to SINAMICS component: <> 0: DO number = 0: Not permitted = 255 DO is only addressed via the logical address
DDSNumber	IN	USINT	O	0	Transfer of the <i>DriveDataSet</i> number for whose <i>EncoderDataSet</i> number the corresponding encoder is to be read in
encoderType	IN	eLMoMaEncoderType	O	ENCODER_INTERFACE_1	Encoder number whose serial number is to be read in
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
commandAborted	OUT	BOOL	-	FALSE	Processing cancelled by a source external to the FB
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 37 Enum for eLMoMaEncoderTypeType

Enum identifier	Value	Comment
ENCODER_INTERFACE_1	1	Encoder interface encoder 1
ENCODER_INTERFACE_2	2	Encoder interface encoder 2
ENCODER_INTERFACE_3	3	Encoder interface encoder 3

3.5 Unit fRWTopology

3.5.1 FBLMoMaReadDqTopology function block

3.5.1.1 General information

Functionality

The **FBLMoMaReadDqTopology** function block is used to read out the actual and target topologies of a SINAMICS S120 drive unit.

The topology information read is prepared by the FB and made available to the user in a corresponding structure of an FB output variable. This data can then be analyzed and manipulated in the SIMOTION device with further functions/function blocks.

The FB thus provides two functions:

- Reading of the target topology of a SINAMICS S120 drive unit
- Reading of the actual topology of a SINAMICS S120 drive unit

3.5.1.2 Schematic diagram in LAD

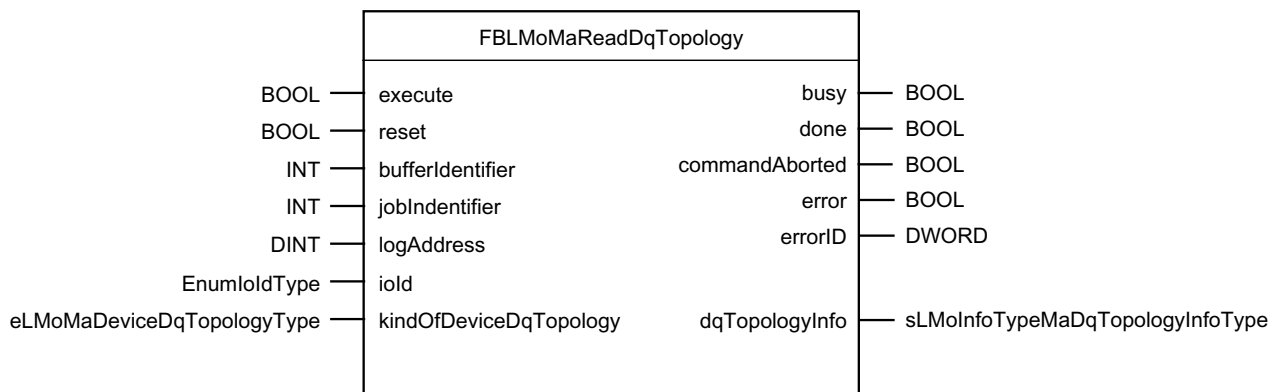


Figure 3-26 Schematic LAD diagram for FBLMoMaReadDqTopology

3.5.1.3 Input and output parameters

Table 3- 38 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop the function block
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCLDPV1ComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of the SINAMICS on which the actual or target topology is to be read out Alternatively, the diagnostics address can be specified.
iold	IN	enumIoldType	O	INPUT	Specification of the data direction of the transferred logical address (input/output)
kindOfDeviceDq-Topology	IN	eLMoMaDeviceDq-TopologyType	O	DEVICE_TARGET_DQ_TOPOLOGY	Enum for the specification of the topology to be read out. DEVICE_TARGET_DQ_TOPOLOGY means the target topology. DEVICE_ACTUAL_DQ_TOPOLOGY means the actual topology.
dqTopologyInfo	OUT	sLMoMaDeviceDqTopologyInfoType	-		Structure for the user to read out topology information
busy	OUT	BOOL	-	FALSE	Function block being processed = TRUE: A job is being processed = FALSE: No jobs pending
done	OUT	BOOL	-	FALSE	Function block successfully completed TRUE: If current job has been completed
commandAborted	OUT	BOOL	-	FALSE	Processing cancelled by a source external to the FB
error	OUT	BOOL	-	FALSE	Function block cancelled with error = TRUE: Job completed with error (see errorID for output parameter) = FALSE: No errors have occurred
errorID	OUT	DWORD	-	16#00000000	Error code The ID can be used to specify an error that has occurred

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 39 Enum for *eLMoMaDeviceDqTopologyType*

Enum identifier	Comment
DEVICE_ACTUAL_DQ_TOPOLOGY	Actual topology
DEVICE_TARGET_DQ_TOPOLOGY	Target topology

Table 3- 40 Output structure *sLMoMaDqTopologyInfoType*

Parameter	File type	Initial value	Description
i16VersionIdentifier	INT	0	Integer value for version code (for internal use only)
u16HeadNodeOfAllComponents	UINT	0	Integer value for version code (for internal use only)
u16NumberOfComponents	UINT	0	Total number of components
asComponent	ARRAY[0..LMOMA_MAX_NUMBER_OF_COMPONENTS - 1] of sLMoMaComponentType		Array of the structure for transferring the data of the individual components

Table 3- 41 Output structure *sLMoMaComponentType*

Parameter	File type	Initial value	Description
u16ComponentNumber	UINT	0	Unique identifier of the component
sgComponentName	STRING[LMOMA_MAX_LENGTH_OF_COMPONENT_NAME]		Name of the component
u16NumberOfDQPorts	UINT	0	Number of DRIVE-CLiQ connections
u16ComponentType	UINT	0	Node identifier component type
eComponentType	eLMoMaClassDeviceTypeType		Component type as enum
u16Manufacturer	UINT	0	Node identifier manufacturer
sgVersion	STRING[1]		Version of the component
sgSerialNumber	STRING[LMOMA_MAX_LENGTH_OF_SERIAL_NUMBER]	0	Node identifier serial number
u16IndexNodeIdentifier	UINT		Node identifier index
sgHWSerialNumber	STRING[LMOMA_MAX_LENGTH_OF_HW_SERIAL_NUMBER]		HW serial number (in the present version of the topology this is identical with <i>sgSerialNumber</i>)
sgPartNumber	STRING[LMOMA_MAX_LENGTH_OF_PART_NUMBER]		MLFB
sgOuterPartNumber	STRING[LMOMA_MAX_LENGTH_OF_OUTER_PART_NUMBER]	0	Outer MLFB
eComparisonLevel	eLMoMaComparisonLevelType	0	Comparison mode of the individual component
u32FWVersion	UDINT		FW version
u32EpromVersion	UDINT		EPROM version

Parameter	File type	Initial value	Description
sgHWVersion	STRING[LMOMA_MAX_LENGTH_OF_HW_VERSION]		HW version
i8NumberODQConnections	SINT		Number of DRIVE-CLiQ connections
asDQConnections	ARRAY[0..LMOMA_MAX_DQ_CONNECTIONS - 1] OF sLMoMaConnectionType		List of all DRIVE-CLiQ connections to components
i8NumberOfOptionSlotConnections	SINT		Number of option slot connections
asOptionSlotConnenction	ARRAY[0..LMOMA_MAX_OPTION_SLOT_CONNECTIONS - 1] OF sLMoMaConnectionType		List of all optional slot connections to components
i8NumberOfPowerConnections	SINT		Number of power connections
asPowerConnections	ARRAY[0..LMOMA_MAX_POWER_CONNECTIONS - 1] OF sLMoMaConnectionType		List of all power connections to components
i8NumberOfAnalogConnections	SINT		Number of analog connections
asAnalogConnections	ARRAY[0..LMOMA_MAX_ANALOG_CONNECTIONS - 1] OF sLMoMaConnectionType		List of all analog connections to components
i8NumberOfMechanicalConnections	SINT		Number of mechanical connections
asMechanicalConnections	ARRAY[0..LMOMA_MAX_MECHANICAL_CONNECTIONS - 1] OF sLMoMaConnectionType		List of all mechanical connections to components

Table 3- 42 Output structure *sLMoMaConnectionType*

Parameter	File type	Initial value	Description
u16FromPort	UINT	0	Connection starting from port number
u16ToComponent	UINT	0	Connection to component with number
u16ToPort	UINT	0	Connection going to port number

Table 3- 43 Enum for *eLMoMaComponentTypeType*

Enum identifier	Comment
UNKNOWN	Unknown (0)
CU	Control unit (3001)
MOTOR_MODULE	Motor module (3002)
LINE_MODULE	Line module (3003)
SENSOR_MODULE	Sensor module (3004)
VOLTAGE_SENSOR_MODULE	VSM (3005)
TERMINAL_MODULE	Terminal module (3006)
HUB	HUB (3007)
OPTION_SLOT	Option slot (3050)
ENCODER	Encoder (3060)
MOTOR	Motor (3070)

Table 3- 44 Enum for *eComparisonLevel*

Enum identifier	Comment
UNKNOWN	Unknown (0)
HIGH	High (16#FF FF)
MEDIUM	Medium (16#0029)
LOW	Low (16#0021)
MINIMUM	Minimum (16#0020)
INDIVIDUAL	Different (16#0004)

Notes on implementing the function block

The reading out of the data field is monitored during readout for simultaneous changes by other sources (a project download via STARTER, for example). If the topology to be read out changes during readout, an error is output and the data is rejected.

3.5.2 Function FCLMoMaGetAllInfosOfSingleComponent

3.5.2.1 General information

Functionality

The **FCLMoMaGetAllInfosOfSingleComponent** function is used to output all topology information for an individual component contained in a target/actual topology that has been read out with the **FBLMoMaReadDqTopology** function block. The corresponding components are addressed using the component numbers.

3.5.2.2 Schematic diagram in LAD

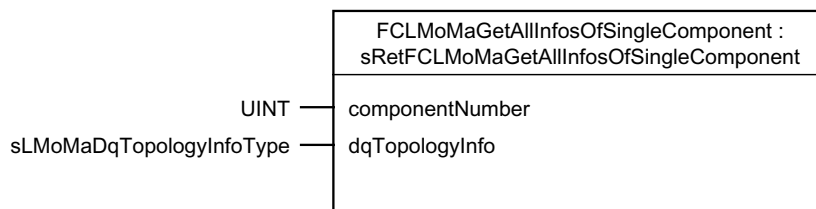


Figure 3-27 Schematic LAD diagram for FCLMoMaGetAllInfosOfSingleComponent

3.5.2.3 Input parameters

Table 3- 45 Input parameters

Parameter	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
componentNumber	IN	UINT	M		Transfer of the component number whose topology information is to be fetched from <i>dqTopologyInfo</i>
dqTopologyInfo	IN/OUT	sLMoMaDqTopologyInfoType	M		Transfer of topology read out

1) Parameter types: IN = input parameter, IN/OUT = IN/OUT parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Return value: *sRetFCLMoMaGetAllInfosOfSingleComponent*

Table 3- 46 Structure of the return value *sRetFCLMoMaGetAllInfosOfSingleComponent*

Parameter	Data type	Meaning
sComponentData	sLMoMaComponentType	Output of the topology information to the required component number
b32errorID	DWORD	The ID can be used to specify an error that has occurred

3.5.3 Function FCLMoMaCompareMLFB

3.5.3.1 General information

Functionality

The **FCLMoMaCompareMLFB** function is used to compare 2 MLFBs in string format. You can use the function to compare MLFBs from the current actual and target topologies for a component. The return value is a BOOL type value and indicates whether or not the 2 MLFBs transferred in string format are identical. The function also checks the *targetMLFB* transfer string for the characters **x** and **?**.

- **X** character: Do not specialize, do not compare, not relevant
- **?** character: Must be specialized, also not relevant

The MLFBs are compared character by character. When compared, the characters must be identical, or the digits with **x** or **?** in the *targetMLFB* are omitted in the corresponding *actualMLFB* during comparison.

3.5.3.2 Schematic diagram in LAD

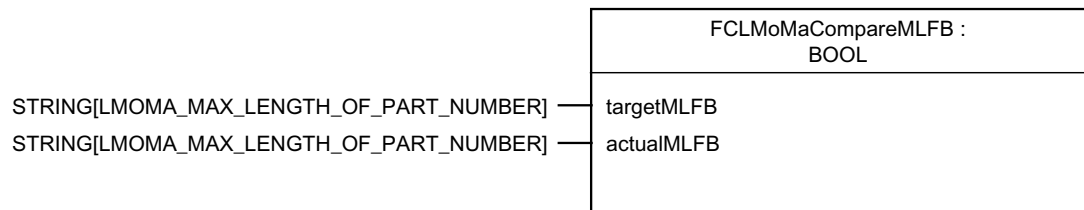


Figure 3-28 Schematic LAD diagram for FCLMoMaCompareMLFB

3.5.3.3 Input parameters

Table 3- 47 Input parameters

Parameter	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
targetMLFB	IN	STRING[LMOMA_MAX_LENGTH_OF_PART_NUMBER]	M	Blank string	Transfer of the MLFB from the target topology
actualMLFB	IN	STRING[LMOMA_MAX_LENGTH_OF_PART_NUMBER]	M	Blank string	Transfer of the MLFB from the actual topology

¹⁾ Parameter types: IN = input parameter

²⁾ Parameter category: M = mandatory parameter, O = optional parameter

3.6 fTopology unit

3.6.1 FBLMoMaGetDOComponents function block

3.6.1.1 General information

Functionality

The **FBLMoMaGetDOComponents** function block is used to identify the components belonging to a drive object. The FB returns the numbers of the components.

3.6.1.2 Schematic diagram in LAD

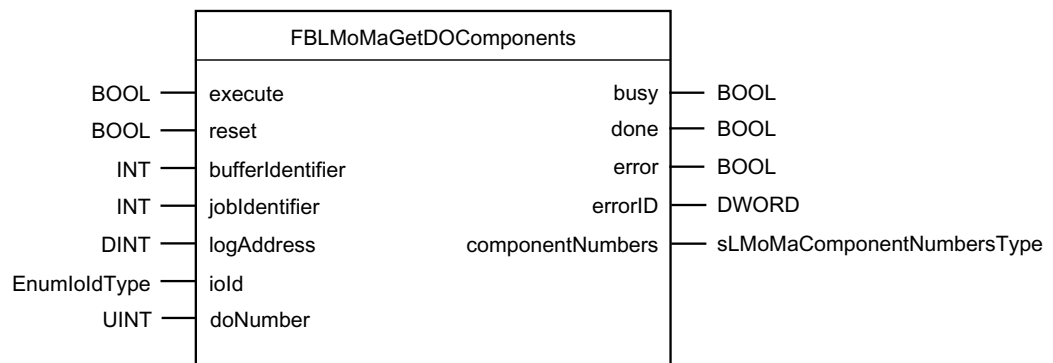


Figure 3-29 Schematic LAD diagram for FBLMoMaGetDOComponents

3.6.1.3 Input and output parameters

Table 3- 48 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop function block and restore original state
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of the SINAMICS on which the FB is to work. Alternatively, the diagnostics address can be specified.
iold	IN	enumIoldType	O	INPUT	Specification of the data direction of the transferred logical address (input/output)
doNumber	IN	UINT	M	0	DO number
componentNumbers	OUT	sLMoMaComponentNumbersType	-	0	Return structure with all component numbers found for the drive object transferred
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 49 Output structure of *sLMoMaComponentNumbersType*

Parameter	Data type	Initial value	Meaning
i16DOType	INT	0	Type of drive object from <i>p107</i>
u8CompNumCU	USINT	0	Component number of the CU (1)
u8NumOfComplnP121	USINT	0	Number of components in the corresponding parameter <i>DOx.p121</i>
au8CompNumP121	ARRAY[0..LMOMA_MAX_NUM_OF_COMP_P121-1] OF USINT	0	Component numbers from <i>DOx.p121</i>
u8NumOfComplnP131	USINT	0	Number of components in the corresponding parameter <i>DOx.p131</i>
au8CompNumP131	ARRAY[0..LMOMA_MAX_NUM_OF_COMP_P131-1] OF USINT	0	Component numbers from <i>DOx.p131</i>
u8NumOfComplnP141	USINT	0	Number of components in the corresponding parameter <i>DOx.p141</i>
au8CompNumP141	ARRAY[0..LMOMA_MAX_NUM_OF_COMP_P141-1] OF USINT	0	Component numbers from <i>DOx.p141</i>
u8NumOfComplnP142	USINT	0	Number of components in the corresponding parameter <i>DOx.p142</i>
au8CompNumP142	ARRAY[0..LMOMA_MAX_NUM_OF_COMP_P142-1] OF USINT	0	Component numbers from <i>DOx.p142</i>
u8NumOfComplnP151	USINT	0	Number of components in the corresponding parameter <i>DOx.p151</i>
au8CompNumP151	ARRAY[0..LMOMA_MAX_NUM_OF_COMP_P151-1] OF USINT	0	Component numbers from <i>DOx.p151</i>
u8CompNumP161	USINT	0	Component numbers from <i>DOx.p161</i>
u8CompNumP162	USINT	0	Component numbers from <i>DOx.p162</i>

Table 3- 50 List of the various drive objects with associated components

Drive object type	Drive object identifier in p0107	Component type	Parameter number	Additional information
SINAMICS S	1	CU	None	Comp. no. is always 1
SINAMICS G	2	CU	None	Comp. no. is always 1
SINAMICS I	3	CU	None	Comp. no. is always 1
SINAMICS CX32	4	CU	None	Comp. no. is always 1
SINAMICS GM	5	CU	None	Comp. no. is always 1
SINAMICS GL	7	CU	None	Comp. no. is always 1
SINAMICS G120	8	CU	None	Comp. no. is always 1
SINAMICS S110	9	CU	None	Comp. no. is always 1
ACTIVE INFEED CONTROL	10	Power unit	DOx.p121	Number of components for power unit is available in <i>DOx.p120</i>
ACTIVE INFEED CONTROLMV	40	VSM	DOx.p141	Number of components for VSM is available in <i>DOx.p140</i>

Drive object type	Drive object identifier in p0107	Component type	Parameter number	Additional information
SMART INFEED CONTROL BASIC INFEED CONTROL BASIC_INFEED CONTROLMV	20 30 41	Power unit	DOx.p121[]	Number of components for power unit is available in <i>DOx.p120</i>
SERVO	11	Power unit Motor Sensor module Encoder	DOx.p121[] DOx.p131[] DOx.p141[] DOx.p142[]	Number of components for power unit is available in <i>DOx.p120</i> Number of components for motor is available in <i>DOx.p130</i> Number of components for sensor module is available in <i>DOx.p140</i> Number of components for sensor module is available in <i>DOx.p140</i>
VECTOR VECTORMV VECTORGL VECTORG120	12 13 14 15	Power unit Motor Sensor module Encoder VSM	DOx.p121[] DOx.p131[] DOx.p141[] DOx.p142[] DOx.p151[]	Number of components for power unit is available in <i>DOx.p120</i> Number of components for motor is available in <i>DOx.p130</i> Number of components for sensor module is available in <i>DOx.p140</i> Number of components for sensor module is available in <i>DOx.p140</i> Number of components for VSM is available in <i>DOx.p150</i>
TB30	100	Option board	DO.p161[0]	
DMC	150	DRIVE-CLiQ HUB	DO.p151[0..1]	There are two components here
TM31 TM41 TM17 TM15 TM54F master TM54F slave	200 201 202 203 205 206	Terminal module	DO.p151[0]	
CU-LINK	254	CU-LINK	DO.p162[0]	

3.6.2 FBLMoMaGetAllExistingDOsOfCU function block

3.6.2.1 General information

Functionality

The **FBLMoMaGetAllExistingDOsOfCU** function block is used to identify all drive objects configured on a control unit.

3.6.2.2 Schematic diagram in LAD

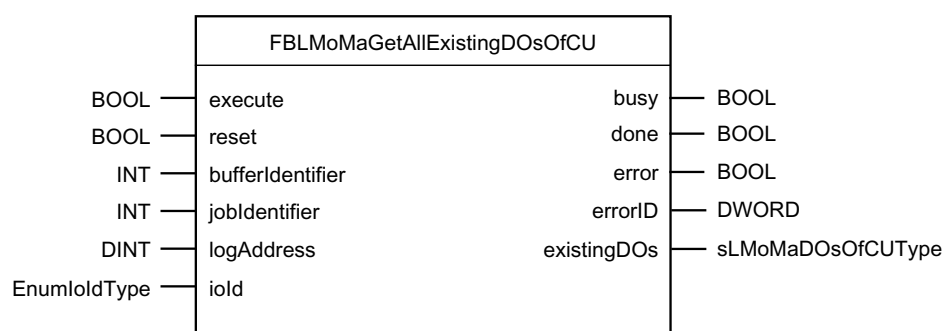


Figure 3-30 Schematic LAD diagram for FBLMoMaGetAllExistingDOsOfCU

3.6.2.3 Input and output parameters

Table 3- 51 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop function block and restore original state
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of the SINAMICS on which the FB is to work. Alternatively, the diagnostics address can be specified.
ioId	IN	enumIoIdType	O	INPUT	Specification of the data direction of the transferred logical address (input/output)
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code
existingDOs	OUT	sLMoMaDOsOfCUType	-		Returns all drive objects found on the CU

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

Table 3- 52 Output structure of *sLMoMaDOsOfCUType*

Parameter	Data type	Initial value	Meaning
sgCUName	STRING[0..255]	0	Name of the control unit
u8NumberOfDOs	USINT	0	Number of configured drive objects
asDriveObjects	ARRAY[0...15] of sLMoMaDOType		Array of the configured drive objects

Table 3- 53 Output structure of *sLMoMaDOType*

Parameter	Data type	Initial value	Meaning
u16DONumber	UINT	0	DO number of the component
sgDOName	STRING[0..255]		Name of the drive object
eDOType	eLMoMaDOType	UNKNOWN	Drive object type

3.7 fDOImage unit

3.7.1 General information

Runtime parameterization of drive objects

Useful auxiliary function blocks for data backup and for setting drive-specific wiring (BiCo between drive objects)

3.7.2 FBLMoMaHandleUnitData function block

3.7.2.1 General information

General information

SIMOTION SCOUT enables the user to save, load or initialize custom data sets. A data set consists of non-retentive or retentive unit variables from the interface or implementation section of a unit.

Data backup is performed by selecting the relevant function.

- **Binary:**
The backed-up data set cannot be read once the version code of the data segment has been changed. Binary data backup is a high-speed method

- In ASCII format:
The backed-up data set can be read once the version code of the data segment has been changed. The values are saved in XML format in a ZIP archive (file name *.dat) on the storage medium (path /USER/SIMOTION/USER_DIR/UPP/UNITDS/<UnitName>/...). This is a rather time-consuming process and requires you to ensure that the symbol information from the unit variables of the interface section is available in the SIMOTION device for the corresponding unit (see Activation: OPC-XML checkbox).
The following data operations are supported:
 - Save a data set (`_saveUnitDataSet` system function). The user can decide whether the data set is to be saved temporarily (RAM disk) or permanently (on MMC with C230-2). There is also the option to overwrite or not overwrite an existing data set
 - Load a data set (`_loadUnitDataSet` system function). This function restores the backed-up values of the unit variables to the interface or implementation section of an ST source file
 - Export a data set (`_exportUnitDataSet` system function). This function exports the values of the unit variables in ASCII format
 - Import a data set (`_importUnitDataSet` system function). This function imports the values (ASCII format) of the unit variables
 - Delete an individual data set (`_deleteUnitDataSet` system function). This function deletes an individual data set containing the backed-up values of the unit variables
 - Check a data set (`_checkExistingUnitDataSet` system function). This function checks whether the specified data set is available on the storage medium with backed-up values of the unit variables
 - Delete all data sets of a unit (`_deleteAllUnitDataSets` system function). This function deletes all data sets with the backed-up values of the unit variables from the interface or implementation section of an ST source file

You can find a more detailed description of the system functions and data backup from the user program in [References 1].

Functionality

The **FBLMoMaHandleUnitData** function block contains all options supported by SIMOTION for working with global unit data. The parameterization of the input interface determines which action is performed with a data set.

The FB primarily consists of a CASE instruction that makes the system functions for data handling from the user program available for selection. A data set is saved, downloaded, etc. dependent upon the parameterization of the *activity* input variable. The corresponding values for the individual actions are listed in the table below. The name of the unit whose unit data is to be used is entered at the FB's *unitName* input (as a STRING between 2 inverted commas). The *dataSetNr* input variable specifies which data set is to be processed. If a data set is to be saved, additional settings can be made using the *storageType* and *overwrite* input variables. The *storageType* variable is used to define where the data set is to be saved.

Default for this *storageType* variable:

- PERMANENT_STORAGE; i.e. the data set is saved to retentive memory and remains intact even after a power failure
- Alternatively, the variable can be set to TEMPORARY_STORAGE. In this case, the data set is saved to the RAM memory and is lost after a power failure

The *overwrite* variable determines whether or not a saved data set can be overwritten.

- TRUE permits overwriting
- FALSE means that the existing data set cannot be overwritten

The *datascope* input variable specifies the section whose unit variables are to be backed up. The *kindOfData* parameter determines whether non-retentive or retentive global variables are to be backed up.

The FB has been programmed for use in a cyclical task (recommendation: BackgroundTask). It can take several task cycles to process the FB (one of the factors determining the processing time is the size of the data set). Another system function, *_getStateOfUnitDataSetCommand*, is used to detect the status (progress) of a processing operation. This function returns the current status of a function for data backup in each task cycle.

The FB is started with a rising edge at the *execute* input. The *done*, *error*, and *errorID* outputs are reset with a falling edge at the *execute* input. However, they retain the **HIGH** state for at least one cycle. The current status of the FB can be read out at these outputs: The *busy* output shows the current processing status of the FB. The *done* output indicates when an action is completed, either with or without error. The *error* output indicates whether or not an error has occurred. This can be specified in greater detail using the *errorID* output parameter.

The *errorID* output parameter can adopt the following values:

Table 3- 54 Values of the errorID output parameter

Value	Meaning
1000	The <i>activity</i> input parameter has been assigned an invalid value.
2 -17	An error has occurred affecting one of the system functions; see also <i>EnumDeviceUnitDataSetCommand</i>
02	Internal error
03	Command failed
04	No free command buffer
05	Command not found
06	Invalid data set ID
07	Error occurred while reading the data set
08	No memory available for writing the data set
09	Missing access right
10	Data set already exists
11	Data set not found
12	Data set unit not found
13	Incorrect data set for the current project
14	Incomplete import of the data
15	No symbol information available
16	Data selection not contained in the data set
17	Data could not be converted

3.7.2.2 Schematic diagram in LAD

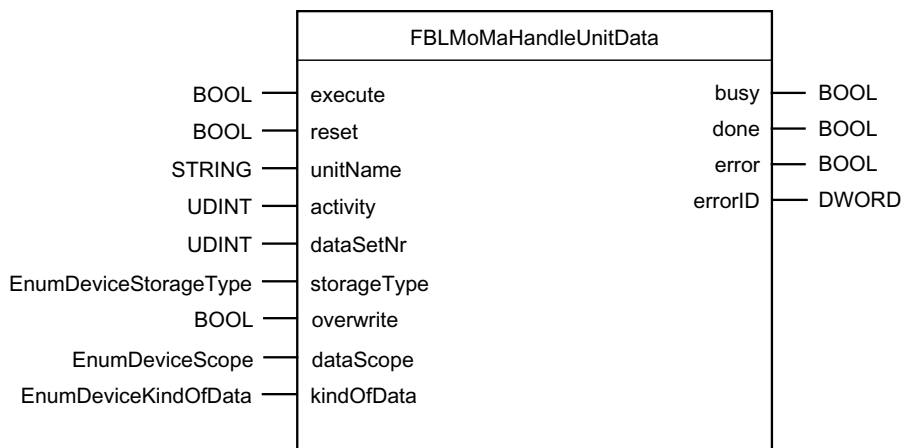


Figure 3-31 Schematic LAD diagram for FBLMoMaHandleUnitData

3.7.2.3 Input and output parameters

Table 3- 55 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop function block and restore original state
unitName	IN	STRING	M		Name of the unit in which the data set is defined
activity	IN	UDINT	M	1	Activity: 1: Save data set 2: Load data set 3: Delete individual data set 4: Check data set 5: Delete all data sets
dataSetNr	IN	UDINT	O	1	Number of the data set to be processed
storageType	IN	EnumDeviceStorageType	O	PermanentStorage	Data set storage type (temporary or permanent)
overwrite	IN	BOOL	O	TRUE	Allow a data set to be overwritten
dataScope	IN	EnumDeviceDataScope	O	_INTERFACE	Section whose unit variables are to be backed up
kindOfData	IN	EnumDeviceKindOfData	O	NO_RETAIN_GLOBAL	Back up retentive or non-retentive global variables
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.7.3 FBLMoMaSetDORamToRom function block

3.7.3.1 General information

Functionality

The **FBLMoMaSetDORamToRom** function block is used to trigger a RAM_TO_ROM for a specific DO. By specifying the DO number 0, or no DO number, a RAM_TO_ROM can also be triggered for the entire device to include all connected drive objects.

Notes on implementing the function block

Please note the following when converting the FB:

- With DO number $\neq 0$, $p971 = 1$ is set
- With DO number = 0, $p977 = 1$ is set
- Done message once *storageProgress* parameter has automatically been reset to 0

3.7.3.2 Schematic diagram in LAD

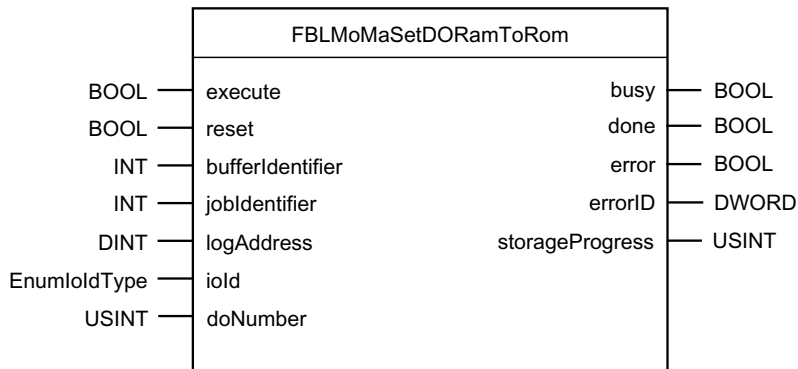


Figure 3-32 Schematic LAD diagram for FBLMoMaSetDORamToRom

3.7.3.3 Input and output parameters

Table 3- 56 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop function block and restore original state
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of the SINAMICS on which the FB is to work. Alternatively, the diagnostics address can be specified.
iold	IN	enumIoldType	O	INPUT	Specification of the data direction of the transferred logical address (input/output)
doNumber	IN	USINT	M	0	DO number of the target drive object
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code
storageProgress	OUT	USINT	-	0	Output of the progress for copying RAM to ROM

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.7.4 FBLMoMaWriteBiCoToDo function block

3.7.4.1 General information

Functionality

The **FBLMoMaWriteBiCoToDo** function block is used to establish a BICO connection to a drive object. This might be necessary because, for example, enable signals are set to 0 by default when a new drive object is created and as a result have to be set manually. Since the internal coding of a BICO connection is rather complex, this FB provides a means of simplifying this functionality. The FB uses the transfer parameters to code the value to be written and writes this automatically to the parameter of the drive object. Similarly, the FB can implement fixed wiring to 0 or 1.

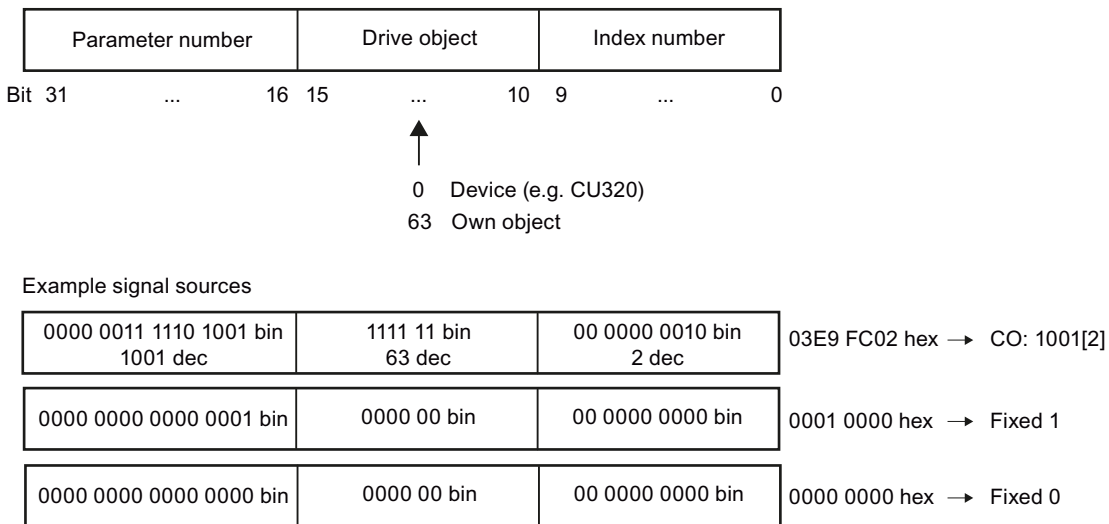


Figure 3-33 Internal coding of BICO connections

Notes on implementing the function block

Please note the following when converting the FB:

- Wiring settings can only be made within a SINAMICS drive
- Cross-device wiring of signals is not possible
- Cross-device wiring of signals from SINAMICS Integrated on SINAMICS CX32 is possible using parameter *CU.p8500*. However, you must follow the instructions in the corresponding SINAMICS documentation

3.7.4.2 Schematic diagram in LAD

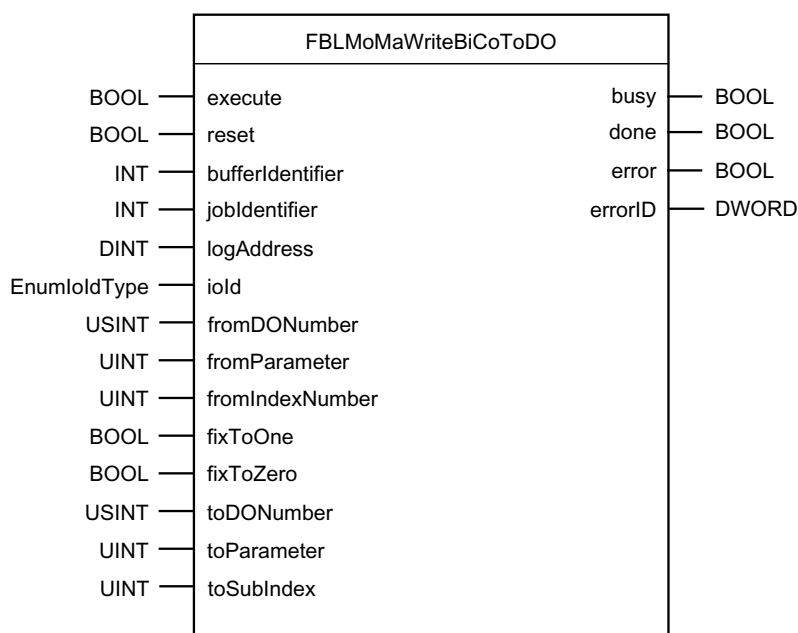


Figure 3-34 Schematic LAD diagram for FBLMoMaWriteBiCoToDo

3.7.4.3 Input and output parameters

Table 3- 57 Input and output parameters

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
execute	IN	BOOL	M	FALSE	Start the functionality
reset	IN	BOOL	O	FALSE	Stop function block and restore original state
bufferIdentifier	IN	INT	O	0	Indexes the job buffer which is to be used for the internal use of the DPV1 service. Buffer management is implemented in the fBuffMana unit in the LDPV1 library.
jobIdentifier	IN	INT	O	0	Unique job identifier for the debug function in buffer management for tracking DPV1 services. See the information about the FCComBufferDiag function in the LDPV1_BufferMana documentation.
logAddress	IN	DINT	M	0	Any logical address of the SINAMICS on which the FB is to work. Alternatively, the diagnostics address can be specified.
iold	IN	enumIoldType	O	INPUT	Specification of the data direction of the transferred logical address (input/output)
fromDONumber	IN	USINT	O	255	Specifies the DO number from which the connection is to be made
fromParameter	IN	UINT	O	0	Parameter number from which the connection is to be made
fromIndexNumber	IN	UINT	O	65535	Index number from which the connection is to be made (subindex or bit number)

Element	P type ¹⁾	Data type	M/O ²⁾	Initial value	Meaning
fixToOne	IN	BOOL	O	FALSE	If TRUE, the connection is set permanently to 1
fixToZero	IN	BOOL	O	FALSE	If TRUE, the connection is set permanently to 0
toDONumber	IN	USINT	O	255	Number of the drive object to which the connection is to be set
toParameter	IN	UINT	M	0	Number of the parameter to which the connection is to be set
toSubIndex	IN	UINT	M	0	Subindex of the parameter in which the connection is to be set
busy	OUT	BOOL	-	FALSE	Function block being processed
done	OUT	BOOL	-	FALSE	Function block successfully completed
error	OUT	BOOL	-	FALSE	Function block cancelled with error
errorID	OUT	DWORD	-	16#00000000	Error code
storageProgress	OUT	USINT	-	0	Output of the progress for copying RAM to ROM

1) Parameter types: IN = input parameter, OUT = output parameter

2) Parameter category: M = mandatory parameter, O = optional parameter

3.8 Constants

Modifiable constants in the cPublic unit of the LMoMa library

Table 3- 58 Modifiable constants in the cPublic unit

Name of the constant	Data type	Description
LMOMA_MAX_DQ_TOPOLOGY_DATA	UINT	Maximum length of the topology information that can be read from a CU
LMOMA_MAX_NUMBER_OF_COMPONENTS	UINT	Maximum number of components of the topology read from a CU

Integration

4.1 Integration in the SIMOTION project

The **LDPV1** library is essential if you wish to use the **LMoMa** library in a SIMOTION project. The current version of the **LDPV1** library is located on the Utilities & Applications storage medium of SIMOTION SCOUT. The **LMoMa** and **LDPV1** libraries are imported into the libraries project tree using the **Import object** function in SIMOTION SCOUT and can subsequently be used in the SIMOTION SCOUT project. To ensure reliable functionality of the function blocks, please use **LDPV1** library buffer management (if you are not doing so already). **LDPV1** library buffer management ensures that conflicts cannot occur during communication on the acyclic communication channel used when using the FBs.

4.2 Required technology objects

Using the LMoMa library with a technology object

The use of the **TP_CAM** technology package (TP) is a minimum requirement if you wish to make full use of the functional scope of the **LMoMa** library. However, the **LMoMa** library can also be used with the **TP_PATH** and **TP_CAM_EXT** technology packages.

Using the LMoMa library without a technology object

However, the **LMoMa** library can also be used without a TP, although in this case its **full functional scope is not available**. All functions and function blocks with which it is possible to transfer a technology object are removed by the compiler during compilation. If the **LMoMa** library is used without a TP, it is essential to also set the **LDPV1** library to use without a TP.

To use both libraries without a TP, proceed as follows: The **cPublic** unit exists in each of the 2 libraries. This unit has no know-how protection and can be changed. Within the unit there is a commented-out define **LDPV1_WITHOUT_TP**. To use the libraries without a TP, simply delete the commented-out elements for this define.

```
{  
  
#define LDPV1_WITHOUT_TP  
  
}
```

The libraries can then also be used in projects without a TP.

Alarm and error messages

5.1 Error messages

The constant definitions corresponding to the error numbers are stored in the **cPublic** unit of the **LDPV1** and **LMoMa** libraries and can be looked up there. The error numbers from *errorId* must always be interpreted in HEX format.

The subindex + 1 of the transferred component of the input array where the error occurred is found in bits 4 - 7 (100xx errors only). If bits 4 - 7 contain a 0, the error occurred previously, when deleting the operation.

Table 5- 1 Possible error messages

Error number	Meaning
0	Error-free
1001	A new <i>execute</i> has been triggered although <i>reset</i> is active.
4110	Internal error: The internal state machine of the FB has an invalid value.
4111	Read job OK. Error affecting read parameters. Check <i>myFBReadDriveMultiParameter.sretReadDriveDriveParameter.parameterResult</i> . Check the error numbers displayed by <i>parameterResult</i> for the <i>readDriveMultiParameter</i> FB.
4112	The control unit addressed is overloaded. Read jobs timeout.
4113	Write job OK. Error affecting parameters to be written. Check <i>myFBWriteDriveMultiParameter.sretWriteDriveDriveParameter.parameterResult</i> . Check the error numbers displayed by <i>parameterResult</i> for the <i>writeDriveMultiParameter</i> FB. The system might have applied a temporary write disable, for example; see parameter <i>CU.p3996</i> write parameters block status.
4114	The control unit addressed is overloaded. Write jobs timeout.
4115	The logical address transferred to the FB is incorrect. Check the logical address, e.g. in HW Config.
4116	The addressed component is signaling a station failure. Check the component.
4117	The component addressed using the logical address is not a SINAMICS component.
4250	The data type of the transferred technology object in the <i>techObject</i> input parameter cannot be evaluated.
4251	The <i>_getStateOfTO</i> system function could not be executed correctly. The error code of the system function is available in the static variable <i>sRetGetStateOfTO.functionresult</i> , which is visible to the user via the symbol browser in the instance of the FBSetActiveInactive FB. The detailed fault analysis from <i>functionResult</i> can be looked up in the system help for <i>_getStateOfTO</i> .
425B	The reading of system variables has failed. The addressed TO might be in <i>Restart</i> or an error might have occurred elsewhere.
4275	System variable cannot be read. <i>TO-Restart</i> might be active.
4353	The data format for the parameter is not valid. Check the input parameters.
4367	Unknown data format of a parameter has been found.
4400	Invalid value for <i>comparisonStage</i> . Only values from 0 to 3 are permissible.
4401	The code number of the device reported by the connected infeed does not match the code number to be set. Compare the <i>modulCode</i> and <i>DOx.r0200</i> inputs.
4402	Parameter <i>doNumberDrive = 0</i> is not permitted.

5.1 Error messages

Error number	Meaning
4403	Transferred addressing of the motor module to be changed is not permitted. Check the values for the unique addressing of the motor module.
4405	At least one transferred motor parameter is 0. However, this is only permitted for <i>motorcoolintypep355</i> .
4406	The connected motor is not an asynchronous motor.
4407	The connected motor is not an asynchronous motor with encoder.
4408	No SMI motor is connected.
4409	<i>technologicalApplication</i> has an invalid value (not 100,101,102).
440A	<i>MotorCode</i> and <i>EncoderCode</i> have not been specified.
440B	<i>MotorCode</i> is invalid.
440C	<i>EncoderCode</i> is invalid.
440D	The connected motor is an SMI (<i>P0302</i> ↔0).
440E	An error occurred while saving the configuration data.
4411	The transferred DDS number is not valid. Check the DDS number.
4412	Drive object is in VIK-NAMUR mode. DDS switchover is not permitted in this mode.
4413	Error affecting the <i>_setAxisSTW</i> system function. See also the <i>errorInfo</i> output and system help for <i>_setAxisSTW</i> .
4414	Data save has been cancelled prematurely. Transferred PDS number is too high. Check the <i>PDSNumber</i> .
4415	Transferred DDS number is too high. Check the <i>DDSNumber</i> .
4416	Encoder to be processed from <i>componentType</i> is not configured.
4417	Parking axis is to be deactivated but is not currently active.
4418	The reference transferred in <i>axis</i> does not belong to an axis.
4419	The axis has been normalized to maximum motor speed.
441A	At least one of the transferred values for the gear ratio is 0.
441B	Timeout on restarting the TO. Restarting the TO takes longer than 10 s. Check the TO.
441C	The nominal velocity is to be taken from the transferred structure, but it is 0 there.
441D	The maximum velocity is to be taken from the transferred structure, but it is 0 there.
441E	The maximum torque is to be taken from the transferred structure, but it is 0 there.
441F	The value for the leadscrew pitch is to be taken from the transferred structure, but it is 0 there.
4420	No axis reference transferred.
4421	Transferred line filter type not permitted.
4422	Transferred optional line filter type not permitted.
4503	Incorrect component number transferred.
4504	No component with the transferred component number found.
4505	The transferred structure is empty.
4506	The constant LMOMA_MAX_LENGTH_OF_COMPONENT_NAME must be less than or equal to LMOMA_MAX_LENGTH_OF_DQ_TOPOLOGY_STRINGS.
4507	The constant LMOMA_MAX_LENGTH_OF_SERIAL_NUMBER must be less than or equal to LMOMA_MAX_LENGTH_OF_DQ_TOPOLOGY_STRINGS.
4508	The constant LMOMA_MAX_LENGTH_OF_HW_SERIAL_NUMBER must be less than or equal to LMOMA_MAX_LENGTH_OF_DQ_TOPOLOGY_STRINGS.
4509	The constant LMOMA_MAX_LENGTH_OF_PART_NUMBER must be less than or equal to LMOMA_MAX_LENGTH_OF_DQ_TOPOLOGY_STRINGS.

Error number	Meaning
450A	The constant LMOMA_MAX_LENGTH_OF_OUTER_PART_NUMBER must be less than or equal to LMOMA_MAX_LENGTH_OF_DQ_TOPOLOGY_STRINGS.
450B	The constant LMOMA_MAX_LENGTH_OF_HW_VERSION must be less than or equal to LMOMA_MAX_LENGTH_OF_DQ_TOPOLOGY_STRINGS.
450C	The constant LMOMA_MAX_LENGTH_OF_FUTURE_FEATURE must be less than or equal to LMOMA_MAX_LENGTH_OF_DQ_TOPOLOGY_STRINGS.
450D	The ID in the topology information for the version code is incorrect. Current version of the topology information cannot be evaluated with this version of the library.
450E	An incorrect ID has been found in the topology information read. Current version of the topology information cannot be evaluated with this version of the library.
450F	Current version of the topology information cannot be evaluated with this version of the library.
4510	Incorrect lengths found in the topology information.
4511	Incorrect identifier found in the topology information.
4512	Unknown version code in the topology information. Current version of the topology information cannot be evaluated with this version of the library.
4513	Unknown topology comparison stage detected in the topology information.
4514	The topology transferred for writing is empty.
4515	The component cannot be switched over to write topology operating mode; see <i>DO.CU.r0002</i> .
4516	The requested change of state is not permitted.
4517	The requested change of state is not permitted with SINAMICS firmware V2.6.2.
4518	The component is switched to internal software error operating mode when the state changes. A power ON is necessary.
4519	The previously valid operating mode has been reactivated automatically.
451A	The control remained in topology error operating mode when the operating mode was changed; e.g. after changing the topology and change of state to READY with an incorrect topology. READY is then never reached.
451B	The actual topology cannot be written. Only the target topology can be written.
451C	The topology information to be read is too voluminous for the memory reserved in the cPublic unit with LMOMA_MAX_DQ_TOPOLOGY_DATA. Constant must be made larger.
451D	When reading the topology information, the length of the topology was changed. The topology read out is invalid.
451E	When reading the topology information, the topology was changed. The topology read out is invalid.
451F	The number of components found exceeds the setting in the MAX_NUMBER_OF_COMPONENTS constant in the cPublic unit.
4520	More components have been found than were transferred in the topology header. Internal error.
4521	The LMOMA_MAX_DQ_CONNECTIONS constant in the dProtected unit is too small. Internal error.
4522	The LMOMA_MAX_OPTION_SLOT_CONNECTIONS constant in the dProtected unit is too small. Internal error.
4523	The LMOMA_MAX_POWER_CONNECTIONS constant in the dProtected unit is too small. Internal error.
4524	The LMOMA_MAX_ANALOG_CONNECTIONS constant in the dProtected unit is too small. Internal error.
4525	The LMOMA_MAX_MECHANICAL_CONNECTIONS constant in the dProtected unit is too small. Internal error.
4526	An unknown connection type has been found in the topology information.
4527	An error has been found in the topology information read (next expected ID is incorrect).
4528	An error has been found in the topology information read (next expected ID with an invalid value).

5.1 Error messages

Error number	Meaning
4529	The actual end and the calculated end of the topology information are not the same.
452B	The number of parameters for the drive object read out where the default setting has been changed is greater than LMOMA_MAX_LENGTH_OF_CHANGED_PARAMETERS in the cPublic unit. Constant must be made larger.
452C	The LMOMA_MAX_DQ_TOPOLOGY_DATA constant in the cPublic unit is too small. The number of values in the parameters for the drive object read out where the default setting has been changed is greater than LMOMA_MAX_LENGTH_OF_CHANGED_VALUES in the cPublic unit. Constant must be made larger.
452D	The drive object to be written and the DO data set do not have the same DO type. Check the input parameters.
452E	The data set to be written at the input of the FB is empty.
452F	The control is not in operating mode 115 Parameter download.
4530	Could not change operating mode. Internal timeout.
4531	An unknown drive object has been found.
4532	An invalid component number has been transferred 1 - 199.
4533	An unknown comparison stage has been transferred (INDIVIDUAL).
4534	The transferred component could not be found in the topology.
4535	An incorrect MLFB has been transferred (<i>partNumber</i>).
4536	An incorrect MLFB has been transferred (<i>outerPartNumber</i>).
4537	An incorrect component type has been transferred.
4538	Comparison stage UNKOWN has been transferred; this is not permitted.
453A	The transferred topology contains fewer than 2 components.
453B	The wiring source component is not present.
453C	The wiring target component is not present.
453D	The transferred components <i>From</i> and <i>To</i> are identical.
453E	The number of the port in <i>fromPort</i> does not exist.
453F	The number of the port in <i>toPort</i> does not exist.
4540	The wiring source port is already in use.
4541	The wiring target port is already in use.
4542	Incorrect port number.
4543	Error affecting values for second component.
4544	Component from second component number does not exist.
4545	The transferred topology cannot be extended. The LMOMA_MAX_NUMBER_OF_COMPONENTS constant in the cPublic unit is too small.
4546	No transferred value received for component number, name, or type.
4547	The transferred component number already exists in the topology.
4548	The transferred component for the header node is not a control unit.
4549	Not all option slot connections on this component are empty.
454A	Not all DRIVE-CLiQ connections on this component are empty.
454B	Not all power connections on this component are empty.
454C	Not all analog connections on this component are empty.
454D	Not all mechanical connections on this component are empty.
454F	The currently set operating mode is incorrect.
4550	The transfer parameters for the source of the connection are incorrect.

Error number	Meaning
100x4	Control unit type components cannot be deleted.
100x5	The component number does not exist.
100x6	Main component cannot be deleted (power unit, TM, TB).
100x7	Component is still assigned; must be disconnected from drive object first.
100x8	Component to be deleted automatically at same time is still assigned.
100x9	CU-LINK drive object is not possible in this environment (e.g. CU320 or CX/NX-CU).
100xA	DO number already noted.

Examples and applications

6.1 Application example

Modules for the application example

The relevant modules for the application example are listed in the following table. Two options are shown. The **configured** modules are the same as the original machine. The modules under **Machine option** correspond to a modified configuration of the original machine which should, however, work with the same machine configuration.

Table 6- 1 Modules of the configured and modified machine

	MLFB	Motor/Encoder/PU code	Configured	Machine option	Comment
Mode of operation					
D445	6AU1445-0AA00-0AA0		Yes		
Infeed					
SLM 36 KW	6SL3130-6TE23-6AB0	10026	Yes		
SLM 16 KW	6SL3130-6TE23-6AB0	10025		Yes	
Module					
18 A/18 A	6SL3120-2TE21-8AA0	10014	Yes		The DRIVE-CLiQ 1 and SMC 20 groß motors are connected on the double motor module.
9 A/9 A	6SL3120-2TE21-0AA0	10013		Yes	
5 A	6SL3120-1TE15-0Axx	10002	Yes		The Asynchron groß motor is connected on the motor module.
3 A	6SL3120-1TE13-0Axx	10001		Yes	
Motors					
DRIVE-CLiQ 1	1FK7043-7AH71-1FG0		Yes		Rated speed 4,500 rpm
DRIVE-CLiQ 2	1FK7022-7AH71-1DA0			Yes	Rated speed 6,000 rpm
Caution: When making changes to DRIVE-CLiQ motors (variance), please note that it will be necessary to run system variables on the associated TO.					
SMC 20 groß	1FK6061-xAF7x-xAxx	23610	Yes		No encoder parameterized

6.1 Application example

	MLFB	Motor/Encoder/PU code	Configured	Machine option	Comment
SMC 20 klein	1FK6042-xAF7x-xAxx	23604		Yes	No encoder parameterized
Asynchroner groß	460 V; 60 Hz; 1.3 kW; 2.7 A; cos φ 0.78; 1690 rpm		Yes		The system variables might also have to be changed at the TO. Please note that motor commissioning also has to be performed or the corresponding parameters set at the DO.
Asynchroner klein	460 V; 60 Hz; 0.21 kW; 0.57 A; cos φ 0.75; 1650 rpm			Yes	

The scenario described above does not correspond to a machine in use. It is used solely to illustrate how the FBs described in this document can be used in practical applications.

Example program

The example program is included in the BackgroundTask of the SIMOTION SCOUT project; it is started once the SIMOTION device has completed ramp-up. When using the FBs described in this document, make sure that the function for writing parameters to drive objects is not disabled. It may be disabled, for example, during first commissioning of a machine. In this case, following successful ramp-up, the SINAMICS module starts by copying data from the actual topology to the target topology. During this operation, writing to parameters is disabled. A reconfiguration of the machine is, therefore, not possible. You can check whether or not this operation is completed during initial commissioning in parameter *p3996* of the relevant SIMOTION device. If parameter *CU.p3996* is set to 0 (writing not disabled), processing of the variance blocks can commence. Any value other than 0 is equivalent to a write disable.

The basic prerequisite for achieving variance by means of application software is to ramp up the SIMOTION device in **Ready to run** operating mode (*CU.r0002 = 10*). Depending on the set comparison stage for the topology, the control unit (CU) might remain in ramp-up with topology error (*CU.r0002 = 33*). In the case of instances of variance within a machine that are processed with the FBs written to the **fDODiversity** unit, the CU can be brought to the end of ramp-up by lowering the topology comparison stage. This can be achieved using the **FBLMoMaSetTopologyComparisonStage** function block.

Table 6-2 Program example

Example code, interface

```

INTERFACE
    USELIB LMoMa, LDPV1;
    PROGRAM exDiversity;
END_INTERFACE
IMPLEMENTATION
//=====
    VAR_GLOBAL
        ginstFBCyclicCheckOfParameterValue : FBLDPV1CyclicCheckOfParameterValue;
        ginstFBCopyRAMToROM : FBLMoMaSetDORAMToROM;
        ginstFBChangeDriveCliqLineModule : FBLMoMaChangeDriveCliqLineModule;
        ginstFBChangeMotorModule : FBLMoMaChangeMotorModule;
        ginstFBChangeAsynchronousMotor : FBLMoMaChangeAsynchronousMotor;
        ginstFBSetInhibitListForMotorCalculation :
FBLMoMaSetInhibitListForMotorCalculation;
        ginstFBResetAndLoadParameterFromCFCard : FBLMoMaResetAndLoadParameterFromCFCard;
        ginstFBChangeMotorWithDriveCliq : FBLMoMaChangeMotorWithDriveCliq;
        ginstFBChangeCatalogSMCMotor : FBLMoMaChangeCatalogSMCMotor;

        gboActivateInhibitList : BOOL := TRUE;
        gsMotorParameter : sMotorParameterType;
        gsinhibitList : sinhibitListType;
        gbErrorID : DWORD;
        // declare the variable gistepper, to control step by step execution
        gistepper : INT := 0;
        giStepAtError : INT;
        giErrorCounter : INT := 0;
        gStorageProgress : USINT;
    END_VAR
//=====
PROGRAM exDiversity
    CASE gistepper OF
        0:
            IF(startupok)
            THEN
                gistepper := 100;
                giStepAtError := 0;
                gbErrorID := 0;
            END_IF;
    
```

Note

Once the SIMOTION SCOUT application software has started up and following successful ramp-up, a cyclic check is first made to ascertain whether parameters can be written.

Table 6-3 Program example

Example code check p3996

```
//=====
// check if CU.p3996 = 0
//=====
100:
  ginstFBCyclicCheckOfParameterValue(execute := TRUE,
    bufferidentifier := 0,
    jobidentifier := 0,
    logaddress := 256,
    ioid := INPUT,
    donumber := 1,
    parameternumber := 3996,
    eformat := UNSIGNED_INTEGER_8,
    valuetocheck := 0,
    cycle := T#1s
  );
IF
  ginstFBCyclicCheckOfParameterValue.Busy = FALSE
THEN
  gbErrorID := ginstFBCyclicCheckOfParameterValue.ErrorID;
  // Save ErrorID
  ginstFBCyclicCheckOfParameterValue(
  // Set execution FALSE
    execute := FALSE
  );
  IF (gbErrorID = 0)
  THEN
    gistepper := 1000; // next step
  ELSE
    giStepAtError := gistepper;
    gistepper := 9999;
  END_IF;
END_IF;
```

Note

Replacement of the configured SLM 36 KW with the SLM 16 KW actually used.

Table 6-4 Program example

Example code change drive cliq line module

```

//=====
// change drive cliq line module
//=====
1000:
  ginstFBChangeDriveCliqLineModule(execute := TRUE,
    bufferIdentifier := 0, // With DPV1-Arbitration
    jobIdentifier := 5,
    // Jobidentifier for buffermanagement
    logAddress := 256,
    ioId := INPUT,
    doNumberDrive := 2,
    moduleCode := 10025,
    disableCalculateReferenzParameter := TRUE
  );
IF
  ginstFBChangeDriveCliqLineModule.Busy = FALSE
THEN
  gbErrorID := ginstFBChangeDriveCliqLineModule.ErrorID;
  // Save ErrorID
  ginstFBChangeDriveCliqLineModule( // Set execution FALSE
    execute := FALSE
  );
  IF (gbErrorID = 0)
  THEN
    gistepper := 3000; // next step
  ELSE
    giStepAtError := gistepper;
    gistepper := 9999;
  END_IF;
END_IF;

```

Note

Replacement of the configured DMM 18 A/18 A with the DMM 9 A/9 A actually used. Recalculation of the associated drive objects is purposely deactivated here, since the motor module parameters belong to the corresponding drive. As the motors associated with the DMM can also be changed hereafter, it would be unnecessary to recalculate the entire drive object several times and, therefore, waste runtime. However, if you only changed 1 of the 2 motors, you would need to perform a recalculation at this stage (this is because either both or none of the associated drive objects are calculated).

Table 6- 5 Program example

Example code change double motor module

```
//=====
// change double motor module
//=====
2000:
  ginstFBChangeMotorModule(execute := TRUE,
    bufferIdentifier := 0, // With DPV1-Arbitration
    jobIdentifier := 5,
    // Jobidentifier for buffermanagement
    logAddress := 256,
    ioId := INPUT,
    doNumberDrive1 := 3,
    doNumberDrive2 := 4,
    moduleCode := 10013,
    disableCalculateReferenzParameter := TRUE,
    disableCalculateDriveObject := FALSE
  );
IF
  ginstFBChangeMotorModule.Busy = FALSE
THEN
  gbErrorID := ginstFBChangeMotorModule.ErrorID;
  // Save ErrorID
  ginstFBChangeMotorModule( // Set execution FALSE
    execute := FALSE
  );
  IF (gbErrorID = 0)
  THEN
    gistepper := 3000; // next step
  ELSE
    giStepAtError := gistepper;
    gistepper := 9999;
  END_IF;
END_IF;
```

Note

Replacement of the configured motor with SMC with the motor with SMC used in the machine option. The same SMC20 module is used for both drives. The associated drive object is recalculated whenever the motor parameters change. Recalculation also includes the section of the modified motor module assigned to this drive.

Table 6- 6 Program example

Example code change smc catalog motor

```

//=====
// change SMC catalog Motor
//=====
3000:
  ginstFBChangeCatalogSMCMotor(execute := TRUE,
    bufferIdentifier := 0, // With DPV1-Arbitration
    jobIdentifier := 5,
    // Jobidentifier for buffermanagement
    logAddress := 256,
    ioId := INPUT,
    doNumberDrive := 3,
    motorCode := 23604,
    encoderCode := 2001,
    disableCalculateReferenzParameter := TRUE,
    technologicalapplication := 101,
    disableCalculateDriveObject := FALSE
  );
IF
  ginstFBChangeCatalogSMCMotor.Busy = FALSE
THEN
  gbErrorID := ginstFBChangeCatalogSMCMotor.ErrorID;
  // Save ErrorID
  ginstFBChangeCatalogSMCMotor( // Set execution FALSE
    execute := FALSE
  );
  IF (gbErrorID = 0)
  THEN
    gistepper := 4000; // next step
  ELSE
    giStepAtError := gistepper;
    gistepper := 9999;
  END_IF;
END_IF;

```

Note

The connected motor with DRIVE-CLiQ is read in again here. As part of this process, the SMI motor connected at this time, and its data, are written to the drive object. The associated drive object is recalculated whenever the motor is read in again. Recalculation also includes the section of the modified motor module assigned to this drive.

Table 6-7 Program example

Example code change smi motor

```
//=====
// change SMI Motor
//=====
4000:
  ginstFBChangeMotorWithDriveCliq(execute := TRUE,
    bufferIdentifier := 0, // With DPV1-Arbitration
    jobIdentifier := 5,
    // Jobidentifier for buffermanagement
    logAddress := 256,
    ioId := INPUT,
    doNumberDrive := 4,
    disableCalculateReferenzParameter := TRUE,
    technologicalapplication := 101,
    disableCalculateDriveObject := FALSE
  );
IF
  ginstFBChangeMotorWithDriveCliq.Busy = FALSE
THEN
  gbErrorID := ginstFBChangeMotorWithDriveCliq.ErrorID;
  // Save ErrorID
  ginstFBChangeMotorWithDriveCliq( // Set execution FALSE
    execute := FALSE
  );
  IF (gbErrorID = 0)
  THEN
    gistepper := 5000; // next step
  ELSE
    giStepAtError := gistepper;
    gistepper := 9999;
  END_IF;
END_IF;
```

Note

Replacement of the configured MM 5 A with the MM 3 A actually used. Recalculation of the associated drive object is purposely deactivated here, since the motor module parameters belong to the motor which is also to be modified. The necessary recalculation is, therefore, made once the motor has been modified.

Table 6-8 Program example

Example code change motor module

```

//=====
// change motor module
//=====
5000:
  ginstFBChangeMotorModule(execute := TRUE,
    bufferIdentifier := 0, // With DPV1-Arbitration
    jobIdentifier := 5,
    // Jobidentifier for buffermanagement
    logAddress := 256,
    ioId := INPUT,
    doNumberDrive1 := 5,
    moduleCode := 10001,
    disableCalculateReferenzParameter := TRUE,
    disableCalculateDriveObject := FALSE
  );
IF
  ginstFBChangeMotorModule.Busy = FALSE
THEN
  gbErrorID := ginstFBChangeMotorModule.ErrorID;
  // Save ErrorID
  ginstFBChangeMotorModule( // Set execution FALSE
    execute := FALSE
  );
  IF (gbErrorID = 0)
  THEN
    gistepper := 6000; // next step
  ELSE
    giStepAtError := gistepper;
    gistepper := 9999;
  END_IF;
END_IF;

```

Note**Step 6000:**

Set the necessary characteristic values of the asynchronous motor from the values on the type plate. The values of this structure are then transferred to the drive object of the asynchronous motor.

Step 6100:

Replacement of the asynchronous motor. The characteristic values of the new motor set in step 6000 are transferred to the drive object. However, in the case of an asynchronous motor, the values of the stationary/turning measurement must either be transferred to the drive object or determined by means of commissioning.

Table 6-9 Program example

Example code change asynchronous motor

```
//=====
// change asynchronous motor
//=====
6000:
  gsmotorparameter.ratedmotorvoltagep304 := 460;
  gsmotorparameter.ratedmotorcurrentp305 := 0.57;
  gsmotorparameter.ratedmotorpowerp307 := 0.21;
  gsmotorparameter.ratedmotorpowerfactorp308 := 0.75;
  gsmotorparameter.ratedmotorfrequencyp310 := 60;
  gsmotorparameter.ratedmotorspeedp311 := 1650;
  gsmotorparameter.maximummotorspeedp322 := 1650;
  gsmotorparameter.motorcoolingtypep335 := 0;
  gsmotorparameter.technologicalapplicationp500 := 102;
  gistepper := 6100;
6100:
  ginstFBChangeAsynchronousMotor(execute := TRUE,
    bufferIdentifier := 0, // With DPV1-Arbitration
    jobIdentifier := 5,
    // Jobidentifier for buffermanagement
    logAddress := 256,
    ioId := INPUT,
    doNumberDrive := 5,
    disableCalculateReferenzParameter := TRUE,
    disableCalculateDriveObject := FALSE,
    motorParameter := gsMotorParameter
  );
  IF
    ginstFBChangeAsynchronousMotor.Busy = FALSE
  THEN
    gbErrorID := ginstFBChangeAsynchronousMotor.ErrorID;
    // Save ErrorID
    ginstFBChangeAsynchronousMotor( // Set execution FALSE
      execute := FALSE
    );
    IF (gbErrorID = 0)
    THEN
      gistepper := 8000; // next step
    ELSE
      giStepAtError := gistepper;
      gistepper := 9999;
    END_IF;
  END_IF;
```

Note

Save all changes from RAM to ROM. This means that the changes made do not have to be made again when the machine is restarted.

Table 6- 10 Program example

Example code copy ram to rom

```

//=====
// copy RAM to ROM
//=====
8000:
  ginstFBCopyRAMToROM(execute := TRUE,
    bufferIdentifier := 0, // With DPV1-Arbitration
    jobIdentifier := 5,
    // Jobidentifier for buffermanagement
    logAddress := 256,
    ioId := INPUT,
    doNumber := 0,
  );
gStorageProgress := ginstFBCopyRAMToROM.StorageProgress;
IF ginstFBCopyRAMToROM.Busy = FALSE
THEN
  gbErrorID := ginstFBCopyRAMToROM.ErrorID;
  // Save ErrorID
  ginstFBCopyRAMToROM( // Set execution FALSE
    execute := FALSE
  );
  IF (gbErrorID = 0)
  THEN
    gistepper := 9000; // next step
  ELSE
    giStepAtError := gistepper;
    gistepper := 9999;
  END_IF;
END_IF;
END_CASE;
END_PROGRAM
END_IMPLEMENTATION

```

Note

Triggering of a warm restart of the corresponding device with loading of the data stored on the storage medium. After restarting, the machine is ready for operation on the drive side. All devices deviating from the configuration have been changed during runtime and can be operated.

Table 6- 11 Program example

Example code Reset and load parameter from CF Card

```
//=====
// Reset and load parameter from CF Card
//=====
9000:
  ginstFBResetAndLoadParameterFromCFCard(execute := TRUE,
    bufferIdentifier := 0, // With DPV1-Arbitration
    jobIdentifier := 5,
    // Jobidentifier for buffermanagement
    logAddress := 256,
    ioId := INPUT,
    );
  IF
    ginstFBResetAndLoadParameterFromCFCard.Busy = FALSE
  THEN
    gbErrorID := ginstFBResetAndLoadParameterFromCFCard.ErrorID; // Save
ErrorID
    ginstFBResetAndLoadParameterFromCFCard(
    // Set execution FALSE
      execute := FALSE
    );
    IF (gbErrorID = 0)
    THEN
      gistepper := 10000; // next step
    ELSE
      giStepAtError := gistepper;
      gistepper := 9999;
    END_IF;
  END_IF;
```

Contacts and Internet addresses

7.1 Contacts

Siemens AG
Industry Sector
I DT MC PM APC
Frauenauracher Strasse 80
D-91056 Erlangen, Germany
Fax: +49 9131 98 1297
Mail to: applications.erlf.aud@siemens.com

7.2 Internet addresses

Additional information on various topics is provided on the following Internet pages.

See also

SIMOTION (www.siemens.com/simotion)
SINAMICS (www.siemens.com/sinamics)
Motion Control / Application Center (www.siemens.com/motioncontrol/apc)
Packaging (www.siemens.com/packaging)

