EcoStruxure Machine Expert

UserMotorTypePlate Library Guide

26.02.2019

EIO0000002682.01

www.schneider-electric.com

Schneider Electric

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

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

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

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

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

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

© 2019 Schneider Electric. All rights reserved.

Table of Contents

Safety Information

Important Information

NOTICE

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

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

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

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

NOTICE

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

PLEASE NOTE

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

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

QUALIFICATION OF PERSONNEL

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

The qualified person must be able to detect possible hazards that may arise from parameterization, modifying parameter values and generally from mechanical, electrical, or electronic equipment. The qualified person must be familiar with the standards, provisions, and regulations for the prevention of industrial accidents, which they must observe when designing and implementing the system.

PROPER USE

This product is a library to be used together with the control systems and servo amplifiers intended solely for the purposes as described in the present documentation as applied in the industrial sector.

Always observe the applicable safety-related instructions, the specified conditions, and the technical data.

Perform a risk evaluation concerning the specific use before using the product. Take protective measures according to the result.

Since the product is used as a part of an overall system, you must ensure the safety of the personnel by means of the design of this overall system (for example, machine design).

Any other use is not intended and may be hazardous.

BEFORE YOU BEGIN

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

⚠ WARNING
UNGUARDED EQUIPMENT ● Do not use this software and related automation equipment on equipment which does not have point-of-operation protection. ● Do not reach into machinery during operation. **Failure to follow these instructions can result in death, serious injury, or equipment damage.**

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

START-UP AND TEST

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check be made and that enough time is allowed to perform complete and satisfactory testing.

⚠ WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

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

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

OPERATION AND ADJUSTMENTS

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer’s instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer’s instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

About the Book

At a Glance

Document Scope

This document describes the functionalities contained in the UserMotorTypePlate library.

Validity Note

This document has been updated for the release of EcoStruxure™ Machine Expert V1.1.

The technical characteristics of the devices described in the present document also appear online. To access the information online:

Step	Action
1	Go to the Schneider Electric home page *www.schneider-electric.com*.
2	In the **Search** box type the reference of a product or the name of a product range. ● Do not include blank spaces in the reference or product range. ● To get information on grouping similar modules, use asterisks (*).
3	If you entered a reference, go to the **Product Datasheets** search results and click on the reference that interests you. If you entered the name of a product range, go to the **Product Ranges** search results and click on the product range that interests you.
4	If more than one reference appears in the **Products** search results, click on the reference that interests you.
5	Depending on the size of your screen, you may need to scroll down to see the data sheet.
6	To save or print a data sheet as a .pdf file, click **Download XXX product datasheet**.

Product Related Information

⚠ WARNING

LOSS OF CONTROL

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

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

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

Before you attempt to provide a solution (machine or process) for a specific application using the POUs found in the library, you must consider, conduct and complete best practices. These practices include, but are not limited to, risk analysis, functional safety, component compatibility, testing and system validation as they relate to this library.

⚠ WARNING

IMPROPER USE OF PROGRAM ORGANIZATION UNITS

- Perform a safety-related analysis for the application and the devices installed.
- Ensure that the Program Organization Units (POUs) are compatible with the devices in the system and have no unintended effects on the proper functioning of the system.
- Use appropriate parameters, especially limit values, and observe machine wear and stop behavior.
- Verify that the sensors and actuators are compatible with the selected POUs.
- Thoroughly test all functions during verification and commissioning in all operation modes.
- Provide independent methods for critical control functions (emergency stop, conditions for limit values being exceeded, etc.) according to a safety-related analysis, respective rules, and regulations.

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Always evaluate the return values when using POUs of a library.

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

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

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

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

Update your application program as required, paying particular attention to I/O address adjustments, whenever you modify the hardware configuration.

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

Incomplete file transfers, such as data files, application files and/or firmware files, may have serious consequences for your machine or controller. If you remove power, or if there is a power outage or communication interruption during a file transfer, your machine may become inoperative, or your application may attempt to operate on a corrupted data file. If an interruption occurs, reattempt the transfer. Be sure to include in your risk analysis the impact of corrupted data files.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION, DATA LOSS, OR FILE CORRUPTION

- Do not interrupt an ongoing data transfer.
- If the transfer is interrupted for any reason, re-initiate the transfer.
- Do not place your machine into service until the file transfer has completed successfully, unless you have accounted for corrupted files in your risk analysis and have taken appropriate steps to prevent any potentially serious consequences due to unsuccessful file transfers.

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

⚠ WARNING

UNINTENDED MOVEMENT OF THE AXIS

- Ensure the proper functioning of the functional safety equipment before commissioning.
- Ensure that you can stop axis movements at any time using functional safety equipment (limit switch, emergency stop) before and during commissioning.

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

⚠ WARNING

UNINTENDED MOVEMENT OF THE SLAVE AXIS

Deactivate the POU that instructs the slave, or disconnect the connection from the master, if the slave axis stops independently from the master.

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

Motion function blocks, except for the Homing function blocks, can only be activated after the mechanical position reference has been established. This is especially important after the start-up of the Sercos motion bus.

⚠ WARNING

INCORRECT HOMING REFERENCE TO MECHANICAL SYSTEM

Ensure that a valid mechanical position reference exists by performing commissioning tests for all operating modes.

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

Related Documents

Title of Documentation	Reference Number
EcoStruxure Machine Expert – Basic Functions and Libraries User Guide	*EIO0000002829 (ENG)*; *EIO0000002830 (FRE)*; *EIO0000002831 (GER)*; *EIO0000002833 (SPA)*; *EIO0000002832 (ITA)*; *EIO0000002834 (CHS)*
Lexium 62 Drive Device Objects and Parameters	*EIO0000003549 (ENG)*
EcoStruxure Machine Expert – Basic 3rd Party Motor Library Guide	*EIO0000003555(ENG)*

You can download these technical publications and other technical information from our website at https://www.schneider-electric.com/en/download

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
IEC 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2015	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2015	Safety of machinery - Emergency stop - Principles for design
IEC 62061:2015	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.

Standard	Description
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2016	Industrial communication networks - Profiles - Part 3: Functional safety fieldbuses - General rules and profile definitions.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive* (*2006/42/EC*) and *ISO 12100:2010*.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

Chapter 1
Presentation of the Library

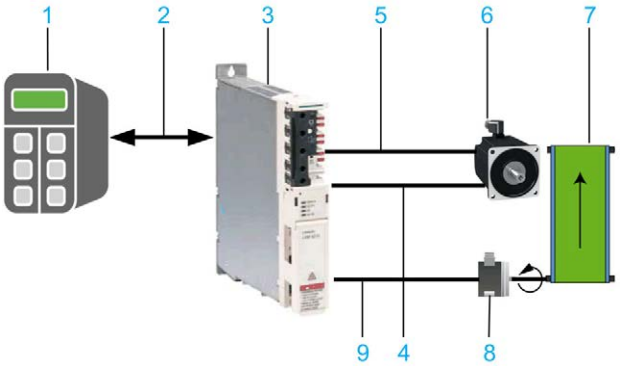
What Is in This Chapter?

This chapter contains the following topics:

General Information

Library Overview

The library UserMotorTypePlate provides functionalities to write the motor typeplate data in the motor encoder or drive and for parameterization of a machine encoder.



1 Lexium Motion Controller (for example, PacDrive LMC)
2 Sercos bus
3 Lexium drive
4 Motor powercable
5 Motor encoder cable (not for BMP or asynchron motors)
6 Motor (optional with motor encoder and Hiperface memory area)
7 Mechanics (for example, conveyor belt)
8 Machine encoder with Hiperface memory area
9 Machine encoder cable

Functional overview of the library

Item	Hardware component	Elements	
1	Lexium motion controller	IEC Area with variables of type: ● `ST_UserMotorData` *(see page 98)* ❍ `ST_UserMotorDataACIM` *(see page 99)* ❍ `ST_UserMotorDataPMSM` *(see page 104)* ● `ST_UserMachineEncoderData` *(see page 96)*	Mass storage (for example flash card) ● .blh ❍ Motor data provided by Schneider Electric ● .mdf ❍ Motor data
3	Drive	Standard components ● Firmware of the drive ● FPGA (Field Programmable Gate Array) ● Typeplate of the drive	Memory area for motor data of type `ST_UserMotorData` *(see page 98)* for connected motor, if data cannot be stored directly in the motor encoder.

Item	Hardware component	Elements
6	Motor with optional encoder	Optional motor encoder Supported `ET_EncoderType` *(see page 30)* elements for motor encoders: ● SinCos Hiperface ● SinCos ● SinCos Hiperface Linear ● SinCos Linear
8	Machine encoder	Supported `ET_EncoderType` *(see page 30)* elements for machine encoders: ● SinCos Hiperface

Preconditions / Restrictions

- To use the machine encoder functionality, a machine encoder with a Hiperface feature and Lexium 62 Advanced Plus drive must be used.
  One of the following preconditions must be met in addition to the Hiperface:
  - ❍ The encoder has a generic typeplate.
  - ❍ The encoder is of type:
    - – SC• 60, SR• 50, SK• 36 (•=S or M)
    - – SEK 37, SEL 37, TTK 70 Schneider Electric version, L 230
- BLH files can be read from a PacDrive LMC controller and be transferred to drive or motor encoder. These files cannot be created by the library, but they are provided by Schneider Electric.
- In case the Hiperface memory area is needed in the motor encoder or the machine encoder, a minimum size of 2 kByte physical memory is needed.

NOTE: If you are using a third-party motor, follow the instructions in the following section.

Typical Use Cases of the Library

- For the use of third-party motors with SinCos encoder:
  - ❍ In case motor encoder has the Hiperface feature, the motor typeplate data can be written to motor encoder.
  - ❍ In case of no Hiperface feature or not enough Hiperface memory, the motor typeplate can be written directly to the drive.
- For the use of third-party motors without motor encoder: the motor typeplate data can be written directly to the drive (for example, asynchron motor).
- For the use of Lexium BMP motor with Lexium 62 Plus or Lexium 62 Advanced Plus by writing .blh file for the motor to the drive.
- For the use of Lexium 62 Advanced Plus with machine encoder with a Hiperface feature and enough Hiperface memory area by writing machine encoder typeplate data to the machine encoder.
- For the use of third-party motors to copy motor encoder data to a file (backup copy).

Using Machine Encoder

The following steps must be taken to be able to use a machine encoder:

Step	Action
1	Create encoder data with `ST_UserMachineEncoderData` of the PacDrive LMC controller. The preparation of data in this structure can be done offline. Only the transmission of data to the drive has to be done in Sercos phase 2.
2	Transfer the encoder data to the machine encoder by using the function block `FB_InitMachineEncoder` when Sercos bus is in phase 2.
3	Restart the Sercos bus (Phase 0 -> Phase 4) to activate the new parameters.

Using Third- Party Motors

The following steps must be taken to be able to use a third-party motor:

Step	Action
1	Create motor data (.mdf) with `ST_MotorData` in the IEC memory area.
2	Use this structure as an input of `FC_MotorDataFileCreate` and create a motor data file to the controller.
3	Set correct value of the parameter `MotorIdentification`: ● Set motor with type plate / 0 if the storage location of the typeplate is the encoder. ● Set motor without type plate / 2 if the storage location of the typeplate is the drive.
4	Verify that Sercos is in phase 2.
5	Call up the function block `FB_MotorDataWrite` with the drive object and the name of the motor data file. **Result:** The function block signals via `q_xDone` that the motor data has been written to the drive. You must set the Sercos to phase 4 in the application. The axis can be used with a third-party motor.

Application examples of third-party motors are described in the chapters Asynchronous Motor *(see EcoStruxure Machine Expert, 3rd Party Motor, Library Guide)* and Linear motor *(see EcoStruxure Machine Expert, 3rd Party Motor, Library Guide)*.

Characteristics of the Library

The table indicates the characteristics of the library:

Characteristic	Value
Library title	UserMotorTypePlate
Company	Schneider Electric
Category	**Application**
Component	**UserMotorTypePlate**

Characteristic	Value
Default namespace	**MTP**
Language model attribute	qualified-access-only *(see EcoStruxure Machine Expert, Functions and Libraries User Guide)*
Forward compatible library	Yes *(see EcoStruxure Machine Expert, Functions and Libraries User Guide)*

NOTE: For this library, qualified-access-only is set. Therefore, the POUs (program organization unit), data structures, enumerations, and constants have to be accessed using the namespace of the library. The default namespace of the library is **MTP**.

Overview of the POUs

Function block / function	Use
`FB_InitMachineEncoder` *(see page 34)*	Reads data of type `ST_UserMachineEncoderData` form the IEC memory area and writes the data to the machine encoder.
`FB_MotorDataDelete` *(see page 45)*	Deletes the motor encoder data from the drive or the motor encoder.
`FB_MotorDataRead` *(see page 51)*	Reads the motor data of the drive or the motor encoder and saves the data as .mdf file on the controller.
`FB_MotorDataWrite` *(see page 56)*	Reads the motor data from the .mdf file of the controller and writes the data to the drive or the motor encoder.
`FB_MotorDataWriteBLH` *(see page 64)*	Reads the motor data from the .blh file of the controller and writes the data to the drive or the motor encoder.
`FB_MotorSerialNumberWrite` *(see page 72)*	Writes the serial number of the motor to the motor typeplate in the drive or the motor encoder.
`FC_EtDiagExtToString` *(see page 78)*	Converts an enumeration element of type `ET_DiagExt` to a string.
`FC_MotorDataFileCreate` *(see page 79)*	Reads data of type `ST_UserMotorData` from IEC memory area and creates a .mdf file on the controller.
`FC_MotorDataFileRead` *(see page 91)*	Reads the .mdf file from the controller and transfers the data to IEC memory area into a variable of type `ST_UserMotorData`.

Overview of the Structures in the Module-Specific Interface

Structure	Use
`ST_UserMachineEncoderData` *(see page 96)*	Contains general machine encoder data.
`ST_UserMotorData` *(see page 98)*	Contains general motor data for all motor types.
`ST_UserMotorDataACIM` *(see page 99)*	Contains motor data, especially for asynchronous motors (ACIM, AC induction motors).
`ST_UserMotorDataPMSM` *(see page 104)*	Contains specific motor data for synchronous motors (PMSM).

Overview of the Enumerations

Enumeration	Use
`ET_DiagExt` *(see page 26)*	Indicates the POU-specific diagnostic and status messages.
`ET_EncoderType` *(see page 29)*	Contains the possible encoder types supported by PacDrive 3 drives.
`ET_MotorType` *(see page 30)*	Contains the possible motor types supported by PacDrive 3 drives.
`ET_StorageLocation` *(see page 31)*	Indicates the possible storage locations (drive or encoder) of the electronic motor typeplate.

Diagnostic Concept

Overview

PacDrive 3 provides a three-layer diagnostic concept for the libraries. This concept is valid for the Technology/Module libraries in the PacDrive 3 system (for example, the library PD_PacDrive.lib) and uses enumerations for diagnostic coding.

In principle, the diagnostic information has the following layers:

1. General information on the exception. No specific knowledge about the POU functionality is necessary.
2. POU-specific diagnostic and status messages (part 1): Detailed information on the source triggering the diagnostic or status messages.
3. POU-specific diagnostic and status messages (part 2): Detailed and dynamic information on the source triggering the diagnostic or status messages.
   This information changes at runtime (for example, information about the condition of the input parameters). This diagnostic output is optional for the POUs.

The diagnostic concept for the PacDrive 3 family of libraries offers the following advantages:

- Online display of the diagnostic messages
- Detailed precision of diagnostic events via the availability of the diagnostic codes
- Overview on the status or exceptional condition of a POU
- Pertinent solutions to correct the causes for exceptional conditions
- Enumerated diagnostic messages to facilitate multi-language support for HMI displays

Structure of the POU-Independent Information

The diagnostic output `q_etDiag` of the type `GD.ET_Diag` provides a library independent diagnostic information, for example `InputParameterInvalid`. The information can suggest a solution for the cause of the diagnostic.

Depending on the value of `GD.ET_Diag`, it is either a status description or an exception message. A value unequal `GD.ET_Diag.Ok` equates to an exception message.

The enumeration `GD.ET_Diag` and its elements are contained in the library PD_GlobalDiagnostics. It also contains a conversion function for the enumeration `GD.ET_Diag`.

The default namespace of the library PD_GlobalDiagnostics is: GD.The POUs, data structures, enumerations, and constants must be addressed using this namespace.

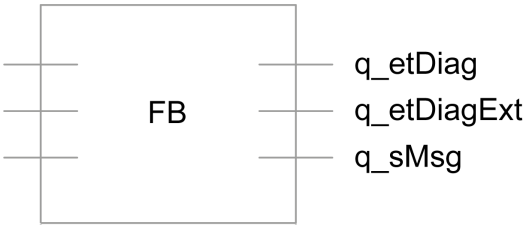
Structure of the POU-Specific Information

The diagnostic information of POUs is designed such that it can express both an exceptional condition or an internal condition (status) during the normal operation of the POU (for example: `WaitForStart`). The information (exceptional condition or status) is reported via the same output (`q_etDiagExt`). The output `q_etDiag` indicates whether a status or an exception is being reported.

Function Blocks

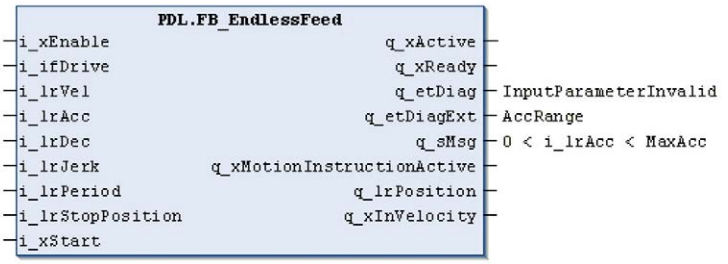
Function blocks have the three outputs `q_etDiag`, `q_etDiagExt`, and optional `q_sMsg`. The outputs are displayed grouped; that is, defined in the POU one after another.

Using schematic POU structure, the following is an example of a function block:



Output	Data type	Meaning
`q_etDiag`	`GD.ET_Diag`	General statement on the diagnostic, for example `InputParameterInvalid`. **NOTE:** If possible, `GD.ET_Diag` contains general formulated diagnostic codes (for example: `DriveConditionInvalid` and `InputParameterInvalid`). Every enumeration element is not only represented by a name, it is also represented by a value. This value can then be used by the HMI so that the translation effort for a neutral language solution of the enumeration name remains maintainable. ● `GD.ET_Diag.Ok`: The status message `q_etDiagExt` gives information about the status of the POU. ● `<> GD.ET_Diag.Ok`: The diagnostic message `q_etDiagExt` gives information about the exception type.
`q_etDiagExt`	`ET_DiagExt`	Extended diagnostic information encoded into a value of the function or service performed within the POU. For example, `AccRange` (Acceleration is out of range)/ `WaitForStart` may be output as a diagnostic or as a status. `q_etDiagExt` provides the numeric value that serves as the index of a more detailed indication of the cause of the output and can be further used as an index into a multi-language set of display messages.
`q_sMsg`	`STRING[80]`	Event triggered optional message that gives more detailed information on the diagnostic condition (for example, `0 < i_lrAcc < MaxAcc`). `q_sMsg` provides a dynamic string containing variable information about the diagnostic in English. `q_sMsg` is modified during runtime. For example, by the exception `VelRange`: `ActualValue: 5003, MaxValue: 5000`. During the normal operation of the POUs (`q_etDiag=GD.ET_Diag.Ok`), `q_sMsg` may provide information concerning the status (for example, the remaining sealing time).

Diagnostic information example:



Functions

The functions also have the outputs `q_etDiag`, `q_etDiagExt`, and the optional `q_sMsg`.

The result of the function is provided by the direct return value in case of one result. If a function has more than one result, then the direct return value is a structure that contains the results.

Alternatively, the results of the function can be returned via several outputs; in this case the direct return value of the function is a BOOL value with a random value that cannot be interpreted.

Read the value `q_etDiag` to ensure that the function was called up successfully. The direct return value of the function does not contain that information.

An exception exists if the return value is <> `GD.ET_Diag.Ok`. The functions are processed completely in the call-up task, then their status can be evaluated and they report via `q_etDiag` and `q_etDiagExt` whether their processing was successful.

Chapter 2
Enumerations

What Is in This Chapter?

This chapter contains the following sections:

Section 2.1
ET_DiagExt

ET_DiagExt - General Information

Overview

Type:	List type
Available as of:	V1.0.0.0

Description

The enumeration ET_DiagExt specifies the POU-specific diagnostic and status messages.

Enumeration Elements

Name	Value	Description
Ok	0	No error detected.
Init	1	Initialization
WaitForSercosPhase2	2	The POU is waiting for Sercos phase 2.
WaitForExecute	3	The POU is waiting for execution.
FunctionNotSupportedByThisDevice	4	The drive does not support the function.
ParameterMotorIdentificationWrongValue	5	The MotorIdentification parameter has the incorrect value.
CheckingMotorDataInDrive	6	The POU verifies whether the drive contains motor data.
SercosCommunicationNotPossible	7	Sercos communication is not possible.
NoMotorDataFoundInDrive	8	No motor data found on the drive.
DeleteMotorDataNotAllowed	9	Deleting motor data is not permitted for motors of Schneider Electric.
Done	10	The command has successfully been performed.
Prepare	11	The POU is preparing the execution.
MotorDataAlreadyStoredInDrive	12	Motor data is already contained in the drive.
MotorDataFileNotFound	13	File not found.
FileAccessNotPossible	14	The file cannot be accessed.
ReinitializationFailed	15	Reinitialization of the drive was not successful.

Name	Value	Description
Disabled	16	The POU is disabled.
DeletingMotorData	17	The motor data in the axis is being deleted.
InputStringTooLarge	18	The entered strings are too long.
NominalSpeedInvalid	19	The rated velocity is invalid.
NominalFrequencyInvalid	20	The rated frequency is invalid.
NominalVoltageInvalid	21	The rated voltage is invalid.
NominalCurrentInvalid	22	The rated current is invalid.
InvalidNumberOfPolePairs	23	The specified pole pair number is invalid.
MotorCosPhiInvalid	24	Cosine phi of the motor is invalid.
RotatingFieldDirectionInvalid	25	The rotary field direction is invalid.
InvalidMotorName	26	The motor name is invalid.
InvalidEncoderType	27	The specified encoder type is invalid.
PeakCurrentInvalid	28	The peak current of the motor is invalid.
ContinuousStallCurrentInvalid	30	The value for the standstill rated current is invalid.
ContinuousStallTorqueInvalid	31	The value for the standstill torque is invalid.
PeakTorqueInvalid	32	The peak torque of the motor is invalid.
PhaseResistanceInvalid	33	The winding resistance is invalid.
QuadraturePhaseInductanceInvalid	34	The inductance value is invalid.
MaxSpeedInvalid	35	The maximum speed of rotation is invalid.
MotorInertiaInvalid	36	The mass inertia of the motor is invalid.
BrakeInvalid	37	The uiBrake value is invalid.
MandatoryParameterInvalid	38	The mandatory parameter is invalid.
MotorTypeNotSupported	39	The motor type is not supported.
CouldNotCreateFile	40	The file for the motor data could not be created.
InvalidStorageLocation	41	The selected storage location for the electronic motor nameplate is invalid.
SingleturnResolutionInvalid	42	The specified singleturn resolution of the encoder is invalid.
ResolutionFineInvalid	43	The specified fine resolution of the encoder is invalid.
ReadingMotorData	44	The motor data is read.
OperationNotAllowed	45	This operation is not permitted for the selected axis.
Executing	46	The POU is being executed.
InvalidDatafield	47	The specified data field number is invalid.

Name	Value	Description
InvalidAddress	48	The specified address is invalid.
NominalPowerInvalid	49	The nominal power is invalid.
EncoderMaxSpeedInvalid	50	The input parameter EncoderMaxSpeed is invalid.
EncoderMaxTempInvalid	51	The input parameter EncoderMaxTemp is invalid.
EncoderTempSensorInvalid	52	The input parameter EncoderTempSensor is invalid.
EncoderNumberOfTurnsInvalid	53	The input parameter EncoderNumberOfTurns is invalid.
EncoderLinesPerRevolutionInvalid	54	The input parameter EncoderLinesPerRevolution is invalid.
SercosNotInPhaseTwo	55	Sercos is not in phase two.
Initializing	56	Initializing
DriveInvalid	57	The connected drive is invalid.
DriveVirtual	58	The connected drive is virtual.

Section 2.2
ET_EncoderType

ET_EncoderType - General Information

Overview

Type:	List type
Available as of:	V1.1.0.0

Description

This enumeration contains all possible encoder types supported by PacDrive 3 drives.

Enumeration Elements

Name	Value	Description
None	INT	The motor has no encoder.
SincosHiperface	INT	The motor has a rotating SinCos / Hiperface encoder, for example, SKS36.
Sincos	INT	The motor has a rotating SinCos encoder (1 Vpp).
SincosHiperfaceLinear	INT	The motor has a linear SinCos Hiperface encoder, for example, TTK70.
SincosLinear	INT	The motor has a linear SinCos encoder (1 Vpp).

Section 2.3
ET_MotorType

ET_MotorType - General Information

Overview

Type:	List type
Available as of:	V1.0.0.0

Description

The enumeration describes the motor type.

Enumeration Elements

Name	Value	Description
RotaryPMSM	INT	Rotary synchronous motor (PMSM)
LinearPMSM	INT	Linear synchronous motor (PMSM)
RotaryACIM	INT	Rotary asynchronous motor (ACIM)

Section 2.4
ET_StorageLocation

ET_StorageLocation - General Information

Overview

Type:	Enumeration type
Available as of:	V1.1.0.0

Description

This enumeration lists all possible storage locations of the electronic motor nameplate.

Enumeration Elements

Name	Value	Description
Encoder	INT	The electronic type plate is stored in the memory of the encoder. The requirement is an encoder with a Hiperface interface and at least 2 kBytes physical memory.
Drive	INT	The electronic type plate is stored in the nonvolatile memory of the device.

Chapter 3
Function Blocks

What Is in This Chapter?

This chapter contains the following sections:

Section 3.1
FB_InitMachineEncoder

FB_InitMachineEncoder - General Information

Overview

Type:	Function block
Available as of:	V1.3.4.0
Inherits from:	–
Implemented:	–

Task

This function block is responsible for initializing the machine encoder with a typeplate. The parameters are defined in `i_stUserMachineEncoderData`.

NOTE: Execute this POU only when required for application needs. The encoder memory area has a limited number of write cycles, frequent execution of the POU can lead to encoder damage.

NOTE: Incorrect machine encoder typeplate data has a direct influence on the behavior of the controller. This function may only be used by technically qualified personnel.

⚠ DANGER

UNINTENDED BEHAVIOR OF THE MOTOR

- Verify assignment of machine encoder data to the motor before writing.
- Ensure that no one is in the operating zone during the startup.

Failure to follow these instructions will result in death or serious injury.

Requirements

Sercos must be in phase 2.

Functional Description

This POU initializes the machine encoder with the parameters which are defined in `i_stUserMachineEncoderData`. The machine encoder data is stored to machine encoder.

The machine encoder data is permanently stored. Thus, the machine encoder data must be written only once.

NOTE:
The machine encoder data must be rewritten when the following applies:

- The machine encoder is replaced
- Parameterization is interrupted (for example, by Sercos diagnostic messages or by power outage)
- The POU `FB_InitMachineEncoder` is not successfully executed (diagnostic messages are issued)

NOTE: The POU has a long processing time (> 1 s). Thus, it must be executed in a task that is not time-critical and with an appropriate watchdog time.

Interface

Input	Data type	Description
`i_xEnable`	BOOL	A rising edge FALSE -> TRUE activates the POU, a falling edge TRUE -> FALSE deactivates the POU. A deactivated POU does not execute any actions.
`i_xExecute`	BOOL	Upon a rising edge of this input. The function block initializes the machine encoder with machine encoder data. While the initialization is running `q_xBusy` = TRUE. After the initialization is completed `q_xDone` = TRUE.
`i_ifDrive`	SystemConfigurationItf.IF_Drive *(see EcoStruxure Machine Expert, SystemConfigurationItf, Library Guide)*	Name of the drive to which the machine encoder is connected.
`i_stUserMachineEncoderData`	ST_UserMachineEncoderData *(see page 96)*	Data structure with machine encoder parameter.

Output	Data type	Description
`q_xActive`	BOOL	TRUE: The POU is active and has to be executed further. FALSE: The POU is inactive.

Output	Data type	Description
q_xReady	BOOL	TRUE: The POU is ready to operate and can accept user commands. FALSE: The function block is not ready to accept user commands.
q_xBusy	BOOL	TRUE: The POU executes the issued user command. FALSE: The POU is waiting for further user commands.
q_xDone	BOOL	TRUE: The user command has been executed successfully. FALSE: The user command is being executed, or none has been issued yet.
q_etDiag	GD.ET_Diag *(see EcoStruxure Machine Expert, PD_GlobalDiagnostics, Library Guide)*	General, library-independent statement on the diagnostic. A value unequal to `GD.ET_Diag.Ok` corresponds to a diagnostic message.
q_etDiagExt	ET_DiagExt *(see page 26)*	POU-specific output on the diagnostic. `q_etDiag = GD.ET_Diag.Ok` -> status message `q_etDiag <> GD.ET_Diag.Ok` -> diagnostic message
q_sMsg	STRING[80]	Event-triggered message which gives more detailed information on the diagnostic state.

Diagnostic Messages

q_etDiag	q_etDiagExt	Enumeration value	Description
OK	Disabled *(see page 37)*	16	The POU is disabled.
OK	Done *(see page 38)*	10	The command has successfully been performed.
OK	Prepare *(see page 42)*	11	The POU is preparing the execution.
OK	WaitForExecute *(see page 43)*	3	The POU is waiting for execution.
OK	Executing *(see page 40)*	46	The POU is being executed.
InputParameterInvalid	DriveInvalid *(see page 38)*	57	The connected drive is invalid.
InputParameterInvalid	EncoderLinesPerRevolutionInvalid *(see page 39)*	54	The input parameter EncoderLinesPerRevolution is invalid.
InputParameterInvalid	EncoderMaxSpeedInvalid *(see page 39)*	50	The input parameter EncoderMaxSpeed is invalid.

q_etDiag	q_etDiagExt	Enumeration value	Description
InputParameterInvalid	EncoderMaxTempInvalid *(see page 39)*	51	The input parameter EncoderMaxTemp is invalid.
InputParameterInvalid	EncoderNumberOfTurns Invalid *(see page 40)*	53	The input parameter EncoderNumberOfTurns is invalid.
InputParameterInvalid	EncoderTempSensorInvalid *(see page 40)*	52	The input parameter EncoderTempSensor is invalid.
InputParameterInvalid	FunctionNotSupportedBy ThisDevice *(see page 41)*	4	The drive does not support the function.
InputParameterInvalid	InvalidEncoderType *(see page 41)*	27	The specified encoder type is invalid.
InputParameterInvalid	InvalidStorageLocation *(see page 42)*	41	The selected storage location for the electronic motor nameplate is invalid.
InputParameterInvalid	MEncInObjectNotAppendedTo Drive *(see page 42)*	59	The MEncIn object is not appended to the drive in the PLC configuration.
InputParameterInvalid	NoValidMachineEncoder Connected *(see page 42)*	60	No valid machine encoder is connected to the hardeware input.
SercosConditionInvalid	SercosCommunicationNot Possible *(see page 43)*	7	Sercos communication is not possible.
SercosConditionInvalid	SercosNotInPhaseTwo *(see page 43)*	55	Sercos is not in phase two.
UnexpectedProgram Behavior	ReinitializationFailed *(see page 43)*	15	Reinitialization of the drive was not successful.

Disabled

Enumeration name:	Disabled
Enumeration value:	16
Description:	The POU is disabled.

Cause	Solution
The POU is disabled.	Set the i_xEnable input from FALSE to TRUE to enable the POUs.

DriveInvalid

Enumeration name:	DriveInvalid
Enumeration value:	57
Description:	The connected drive is invalid.

Cause	Solution
No valid drive was applied at i_ifDrive of FB_InitMachineEncoder	A valid drive must be transferred to the i_ifDrive.

DriveVirtual

Enumeration name:	DriveVirtual
Enumeration value:	58
Description:	The connected drive is virtual.

Cause	Solution
The WorkingState of the connected drive is not real.	Set WorkingMode of the connected drive to real.

Done

Enumeration name:	Done
Enumeration value:	10
Description:	The command has successfully been performed.

The motor data has been deleted successfully. New motor data may be written to the drive.

EncoderLinesPerRevolutionInvalid

Enumeration name:	EncoderLinesPerRevolutionInvalid
Enumeration value:	54
Description:	The input parameter EncoderLinesPerRevolution is invalid.

Cause	Solution
An invalid value was transferred at the ST_UserMachineEncoderData.uiEncoderLinesPerRevolution input.	A value between MTP.Gc_uiEncoderLinesPerRevolutionMinValue and MTP.Gc_uiEncoderLinesPerRevolutionMaxValue must be transferred to the input.

EncoderMaxSpeedInvalid

Enumeration name:	EncoderMaxSpeedInvalid
Enumeration value:	50
Description:	The input parameter EncoderMaxSpeed is invalid.

Cause	Solution
An invalid value was transferred at the ST_UserMachineEncoderData.uiEncoderMaxSpeed input.	A value between MTP.Gc_uiEncoderMaxSpeedMinValue and MTP.Gc_uiEncoderMaxSpeedMaxValue must be transferred to the input.

EncoderMaxTempInvalid

Enumeration name:	EncoderMaxTempInvalid
Enumeration value:	51
Description:	The input parameter EncoderMaxTemp is invalid.

Cause	Solution
An invalid value was transferred at the ST_UserMachineEncoderData.uiEncoderMaxTemp input.	A value between MTP.Gc_uiEncoderMaxTempMinValue and MTP.Gc_uiEncoderMaxTempMaxValue must be transferred to the input.

EncoderNumberOfTurnsInvalid

Enumeration name:	EncoderNumberOfTurnsInvalid
Enumeration value:	53
Description:	The input parameter EncoderNumberOfTurns is invalid.

Cause	Solution
An invalid value was transferred at the ST_UserMachineEncoderData.uiEncoderNumberOfTurns input.	A value between MTP.Gc_uiEncoderNumberOfTurnsMinValue and MTP.uiEncoderNumberOfTurnsMaxValue must be transferred to the input.

EncoderTempSensorInvalid

Enumeration name:	EncoderTempSensorInvalid
Enumeration value:	52
Description:	The input parameter EncoderTempSensor is invalid.

Cause	Solution
An invalid value was transferred at the ST_UserMachineEncoderData.uiEncoderTempSensor input.	The value 0: no temperature sensor or 1: with temperature sensor must be transferred to the input.

Executing

Enumeration name:	Executing
Enumeration value:	46
Description:	The POU is being executed.

The POU is actually in the executing phase.

FunctionNotSupportedByThisDevice

Enumeration name:	FunctionNotSupportedByThisDevice
Enumeration value:	4
Description:	The drive does not support the function.

Cause	Solution
Invalid object type	Verify the selected object.

InvalidEncoderType

Enumeration name:	InvalidEncoderType
Enumeration value:	27
Description:	The specified encoder type is invalid.

Cause	Solution
An invalid value was transferred at the ST_UserMachineEncoderData.etEncoderType input.	The value ET_EncoderType.SincosHiperface must be transferred to the input.

InvalidStorageLocation

Enumeration name:	InvalidStorageLocation
Enumeration value:	41
Description:	The selected storage location for the electronic motor nameplate is invalid.

Cause	Solution
There is no Hiperface encoder connected to the physical machine encoder input.	Connect a Hiperface encoder to the physical machine encoder input.
The encoder does not provide sufficient memory or there is no encoder.	Connect an encoder with Hiperface interface with a minimum of 2 kBytes physical memory.

MEncInObjectNotAppendedToDrive

Enumeration name:	MEncInObjectNotAppendedToDrive
Enumeration value:	59
Description:	The MEncIn object is not appended to the drive in the PLC configuration.

Cause	Solution
There is no MEncIn object appended to the drive.	Append the MEncIn object to the drive in the PLC configuration.
The drive in PLC configuration does not support a machine encoder object.	Set the drive in PLC configuration to Lexium 62 Advanced Plus and append the MEncIn object to the drive in the PLC configuration.

NoValidMachineEncoderConnected

Enumeration name:	NoValidMachineEncoderConnected
Enumeration value:	60
Description:	No valid machine encoder is connected to the hardeware input.

Cause	Solution
There is no SinCos Hiperface encoder connected to the machine encoder hardware input.	Connect a SinCos Hiperface encoder to the machine encoder hardware input.

Prepare

Enumeration name:	Prepare
Enumeration value:	11
Description:	The POU is preparing the execution.

The POU in the q_xBusy execution is TRUE.

ReinitializationFailed

Enumeration name:	ReinitializationFailed
Enumeration value:	15
Description:	Reinitialization of the drive was not successful.

Cause	Solution
Reinitialization was not successful.	Verify the Sercos phase.

SercosCommunicationNotPossible

Enumeration name:	SercosCommunicationNotPossible
Enumeration value:	7
Description:	Sercos communication is not possible.

Cause	Solution
Sercos communication is not possible.	Verify the Sercos phase and the wiring.

SercosNotInPhaseTwo

Enumeration name:	SercosNotInPhaseTwo
Enumeration value:	55
Description:	Sercos is not in phase two.

Cause	Solution
Sercos communication is not possible. The Sercos needs to be in phase two to be able to write to the machine encoder.	Set Sercos to phase two.

WaitForExecute

Enumeration name:	WaitForExecute
Enumeration value:	3
Description:	The POU is waiting for execution.

The POU is active and ready for execution. q_xReady is TRUE.

Examples

```
VAR
    fbInitMachineEncoder     : MTP.FB_InitMachineEncoder;
    stUserMachineEncoderData : MTP.ST_UserMachineEncoderData;
    xInitMachineEncoderData  : BOOL := TRUE;
    xEnable                  : BOOL;
    xExecute                 : BOOL;
END_VAR
```

```
IF xInitMachineEncoderData THEN
    stUserMachineEncoderData.etEncoderType := MTP.ET_EncoderType.Sincos
Hiperface;
    stUserMachineEncoderData.uiEncoderMaxSpeed           := 12000;
    stUserMachineEncoderData.uiEncoderMaxTemp            := 130;
    stUserMachineEncoderData.uiEncoderTempSensor         := 1;
    stUserMachineEncoderData.uiEncoderNumberOfTurns      := 4096;
    stUserMachineEncoderData.uiEncoderLinesPerRevolution := 128;
END_IF
```

```
fbInitMachineEncoder(
    i_xEnable                   := xEnable,
    i_xExecute                  := xExecute,
    i_ifDrive                  := DRV_Lexium62AdvancedPlus,
    i_stUserMachineEncoderData := stUserMachineEncoderData,
   );
```

Section 3.2
FB_MotorDataDelete

FB_MotorDataDelete - General Information

Overview

Type:	Function block
Available as of:	V1.0.0.0
Inherits from:	–
Implemented:	–

Task

This function block deletes motor data in the drive or motor encoder, depending on the selected `i_etStorageLocation`.

NOTE: Execute this POU only when required for application needs. The encoder memory area has a limited number of write cycles, frequent execution of the POU can lead to encoder damage.

Requirements

Sercos must be in phase 2.

Description

This function block deletes the motor data in the selected storage location. After the deletion of the old motor data, the new motor data can be written to the axis using `FB_MotorDataWrite` *(see page 56)*.

NOTE: The POU has a long processing time (> 1 s). Thus, it must be executed in a task that is not time-critical and with an appropriate watchdog time.

Interface

Input	Data type	Description
i_xEnable	BOOL	A rising edge FALSE -> TRUE activates the POU, a falling edge TRUE -> FALSE deactivates the POU. A deactivated POU does not execute any actions.
i_xExecute	BOOL	FALSE -> TRUE: The function block deletes the motor data in the selected axis. While the deletion is running q_xBusy = TRUE. After the deletion is completed q_xDone changes to TRUE.
i_ifDrive	SystemConfigurationItf.IF_Drive *(see EcoStruxure Machine Expert, SystemConfigurationItf, Library Guide)*	Input for the axis that shall be controlled.
i_etStorageLocation	ET_StorageLocation *(see page 31)*	Storage location from which the motor data is read.

Output	Data type	Description
q_xActive	BOOL	TRUE: The POU is active and has to be executed further. FALSE: The POU is inactive.
q_xReady	BOOL	The POU is ready to delete the motor data. Verify whether the requirements are met if the state is FALSE.
q_xBusy	BOOL	TRUE: The POU executes the issued user command (deleting motor data in the drive). FALSE: The POU is waiting for further user commands.
q_xDone	BOOL	TRUE: The user command (delete motor data in the drive) has been executed. Using FB_MotorDataWrite, new motor data can be programmed. FALSE: The user command is being executed, or none has been issued yet.
q_etDiag	GD.ET_Diag *(see EcoStruxure Machine Expert, PD_GlobalDiagnostics, Library Guide)*	General, library-independent statement on the diagnostic. A value unequal to GD.ET_Diag.Ok corresponds to a diagnostic message.
q_etDiagExt	ET_DiagExt *(see page 26)*	POU-specific output on the diagnostic. q_etDiag = GD.ET_Diag.Ok -> status message q_etDiag <> GD.ET_Diag.Ok -> diagnostic message
q_sMsg	STRING[80]	Event-triggered message which gives more detailed information on the diagnostic state.

Diagnostic Messages

q_etDiag	q_etDiagExt	Enumeration value	Description
OK	CheckingMotorDataInDrive *(see page 47)*	6	The POU verifies whether the drive contains motor data.
OK	DeletingMotorData *(see page 48)*	17	The motor data in the axis is being deleted.
OK	Disabled *(see page 48)*	16	The POU is disabled.
OK	Done *(see page 48)*	10	The command has successfully been performed.
OK	WaitForExecute *(see page 50)*	3	The POU is waiting for execution.
OK	WaitForSercosPhase2 *(see page 50)*	2	The POU is waiting for Sercos phase 2.
DriveConditionInvalid	DeleteMotorDataNotAllowed *(see page 48)*	9	Deleting motor data is not permitted for motors of Schneider Electric.
DriveConditionInvalid	NoMotorDataFoundInDrive *(see page 49)*	8	No motor data found on the drive.
InputParameterInvalid	FunctionNotSupportedByThisDevice *(see page 49)*	4	The drive does not support the function.
InputParameterInvalid	InvalidStorageLocation *(see page 49)*	41	The selected storage location for the electronic motor nameplate is invalid.
SercosConditionInvalid	SercosCommunicationNotPossible *(see page 50)*	7	Sercos communication is not possible.

CheckingMotorDataInDrive

Enumeration name:	CheckingMotorDataInDrive
Enumeration value:	6
Description:	The POU verifies whether the drive contains motor data.

The function block is being executed. Waiting until q_xDone has the value TRUE.

DeleteMotorDataNotAllowed

Enumeration name:	DeleteMotorDataNotAllowed
Enumeration value:	9
Description:	Deleting motor data is not permitted for motors of Schneider Electric.

Cause	Solution
The motor data in the selected axes must not be deleted.	Verify the selected axis.

DeletingMotorData

Enumeration name:	DeletingMotorData
Enumeration value:	17
Description:	The motor data in the axis is being deleted.

The function block is being executed. Waiting until q_xDone has the value TRUE.

Disabled

Enumeration name:	Disabled
Enumeration value:	16
Description:	The POU is disabled.

Cause	Solution
The POU is disabled.	Set the i_xEnable input from FALSE to TRUE to enable the POUs.

Done

Enumeration name:	Done
Enumeration value:	10
Description:	The command has successfully been performed.

The motor data has been deleted successfully. New motor data may be written to the drive.

FunctionNotSupportedByThisDevice

Enumeration name:	FunctionNotSupportedByThisDevice
Enumeration value:	4
Description:	The drive does not support the function.

Cause	Solution
Invalid object type	Verify the selected object.

InvalidStorageLocation

Enumeration name:	InvalidStorageLocation
Enumeration value:	41
Description:	The selected storage location for the electronic motor nameplate is invalid.

Cause	Solution
There is no Hiperface encoder connected to the physical machine encoder input.	Connect a Hiperface encoder to the physical machine encoder input.
The encoder does not provide sufficient memory or there is no encoder.	Connect an encoder with Hiperface interface with a minimum of 2 kBytes physical memory.

NoMotorDataFoundInDrive

Enumeration name:	NoMotorDataFoundInDrive
Enumeration value:	8
Description:	No motor data found on the drive.

Cause	Solution
No motor data is stored in the selected axis.	Verify the selected axis.

SercosCommunicationNotPossible

Enumeration name:	SercosCommunicationNotPossible
Enumeration value:	7
Description:	Sercos communication is not possible.

Cause	Solution
Sercos communication is not possible.	Verify the Sercos phase and the wiring.

WaitForExecute

Enumeration name:	WaitForExecute
Enumeration value:	3
Description:	The POU is waiting for execution.

The POU is active and ready for execution. q_xReady is TRUE.

WaitForSercosPhase2

Enumeration name:	WaitForSercosPhase2
Enumeration value:	2
Description:	The POU is waiting for Sercos phase 2.

The POU is active and is waiting for Sercos phase 2. q_xReady is FALSE.

Section 3.3
FB_MotorDataRead

FB_MotorDataRead - General Information

Overview

Type:	Function block
Available as of:	V1.1.0.0
Inherits from:	–
Implemented:	–

Task

Reading the motor data and writing the motor data to a binary file.

Description

The motor data of a selected axis is read and then written to a binary file. You can define the file name. The binary file can be converted into a `ST_UserMotorData` data structure using the `FC_MotorDataRead` function.

Interface

Input	Data type	Description
i_xEnable	BOOL	A rising edge FALSE -> TRUE activates the POU, a falling edge TRUE -> FALSE deactivates the POU. A deactivated POU does not execute any actions.
i_xExecute	BOOL	FALSE -> TRUE: The POU reads the motor data from the selected axis and writes it to a binary file. During this process, q_xBusy = TRUE. As soon as actions are completed, q_xDone changes to TRUE.
i_ifDrive	SystemConfigurationItf.IF_Drive *(see EcoStruxure Machine Expert, SystemConfigurationItf, Library Guide)*	Input for the axis that shall be controlled.
i_sFilename	STRING	Name of the file in which the motor data shall be written. The file is automatically created by the function block.
i_etStorageLocation	ET_StorageLocation *(see page 31)*	Storage location from which the motor data is read.

Output	Data type	Description
q_xActive	BOOL	TRUE: The POU is active and has to be executed further. FALSE: The POU is inactive.
q_xReady	BOOL	TRUE: The POU is ready to operate and can accept user commands. FALSE: The function block is not ready to accept user commands.
q_xBusy	BOOL	TRUE: The POU executes the issued user command. FALSE: The POU is waiting for further user commands.
q_xDone	BOOL	TRUE: The user command has been executed successfully. FALSE: The user command is being executed, or none has been issued yet.
q_etDiag	GD.ET_Diag *(see EcoStruxure Machine Expert, PD_GlobalDiagnostics, Library Guide)*	General, library-independent statement on the diagnostic. A value unequal to GD.ET_Diag.Ok corresponds to a diagnostic message.

Output	Data type	Description
q_etDiagExt	ET_DiagExt *(see page 26)*	POU-specific output on the diagnostic. q_etDiag = GD.ET_Diag.Ok -> status message q_etDiag <> GD.ET_Diag.Ok -> diagnostic message
q_sMsg	STRING[80]	Event-triggered message which gives more detailed information on the diagnostic state.

Diagnostic Messages

q_etDiag	q_etDiagExt	Enumeration value	Description
OK	Disabled *(see page 54)*	16	The POU is disabled.
OK	Done *(see page 54)*	10	The command has successfully been performed.
OK	ReadingMotorData *(see page 54)*	44	The motor data is read.
OK	WaitForExecute *(see page 55)*	3	The POU is waiting for execution.
OK	WaitForSercosPhase2 *(see page 55)*	2	The POU is waiting for Sercos phase 2.
FileHandlingInvalid	CouldNotCreateFile *(see page 53)*	40	The file for the motor data could not be created.
SercosConditionInvalid	SercosCommunicationNotPossible *(see page 54)*	7	Sercos communication is not possible.

CouldNotCreateFile

Enumeration name:	CouldNotCreateFile
Enumeration value:	40
Description:	The file for the motor data could not be created.

Cause	Solution
There already is a file with the file name i_sFilename.	Use a different file name.
The flash disk of the controller has no more free memory.	Remove files from the flash disk that are not used. Use a flash disk with more memory.

Disabled

Enumeration name:	Disabled
Enumeration value:	16
Description:	The POU is disabled.

Cause	Solution
The POU is disabled.	Set the i_xEnable input from FALSE to TRUE to enable the POUs.

Done

Enumeration name:	Done
Enumeration value:	10
Description:	The command has successfully been performed.

The motor data has been successfully read. A file with the file name i_sFilename has been created.

ReadingMotorData

Enumeration name:	ReadingMotorData
Enumeration value:	44
Description:	The motor data is read.

The function block is being executed. Waiting until q_xDone has the value TRUE.

SercosCommunicationNotPossible

Enumeration name:	SercosCommunicationNotPossible
Enumeration value:	7
Description:	Sercos communication is not possible.

Cause	Solution
Sercos communication is not possible.	Verify the Sercos phase and the wiring of the Sercos devices.

WaitForExecute

Enumeration name:	WaitForExecute
Enumeration value:	3
Description:	The POU is waiting for execution.

The POU is active and ready for execution. q_xReady is TRUE.

WaitForSercosPhase2

Enumeration name:	WaitForSercosPhase2
Enumeration value:	2
Description:	The POU is waiting for Sercos phase 2.

The POU is active and is waiting for Sercos phase 2. q_xReady is FALSE.

Section 3.4
FB_MotorDataWrite

FB_MotorDataWrite - General Information

Overview

Type:	Function block
Available as of:	V1.0.0.0
Inherits from:	–
Implements:	–

Task

This function block writes the motor data to the selected storage location. Also refer to the chapter Asynchronous Motor *(see EcoStruxure Machine Expert, 3rd Party Motor, Library Guide)*.

NOTE: Execute this POU only when required for application needs. The encoder memory area has a limited number of write cycles, frequent execution of the POU can lead to encoder damage.

NOTE: Incorrect machine encoder typeplate data has a direct influence on the behavior of the controller. This function may only be used by technically qualified personnel.

⚠ DANGER

UNINTENDED BEHAVIOR OF THE MOTOR

- Verify assignment of machine encoder data to the motor before writing.
- Ensure that no one is in the operating zone during the startup.

Failure to follow these instructions will result in death or serious injury.

NOTE: A deactivated monitoring function can lead to a possible undetected overheating of the motor.

⚠ DANGER

FIRE DUE TO OVERHEATING OF THE MOTOR

- Verify assignment of motor data to the motor before writing.
- Use appropriate methods to verify thermal overload of the motor.
- Ensure that no one is in the operating zone during the startup.

Failure to follow these instructions will result in death or serious injury.

Requirements

- Sercos must be in phase 2.
- No motor data must be stored in the selected drive (use `FB_MotorDataDelete`, if necessary).
- The `MotorIdentification` *(see Lexium LXM62 Drive, Device Objects and Parameters,)* parameter of the axis must be set to the correct value.

Description

This POU receives a file with motor data and writes it into the selected axis. Beforehand the file has to be created with `FC_MotorDataFileCreate` *(see page 79)*.

The motor data is stored to the selected storage location.

- In the memory of the encoder, if sufficient memory is available in the encoder.
- In the servo amplifier, if the encoder has no or insufficient memory or no encoder is used.

The motor data is permanently stored. Thus, the motor data must be written only once.

NOTE: When replacing drives or motors, make sure that the correct motor data is stored in the drive or motor. Under certain circumstances, the motor data must be rewritten.

If motor is to be modified (for example, because the motor was replaced), you must first delete the existing motor data with `FB_MotorDataDelete` *(see page 45)*. Motor data must not be overwritten.

NOTE: The POU has a long processing time (> 1 s). Thus, it must be executed in a task that is not time-critical and with an appropriate watchdog time.

Interface

Input	Data type	Description
`i_xEnable`	BOOL	A rising edge FALSE -> TRUE activates the POU, a falling edge TRUE -> FALSE deactivates the POU. A deactivated POU does not execute any actions.

Input	Data type	Description
i_xExecute	BOOL	FALSE -> TRUE: The function block writes the motor data into the selected axis. During the writing procedure q_xBusy = TRUE. As soon as the writing procedure is completed q_xDone changes to TRUE.
i_ifDrive	SystemConfigurationItf.IF_Drive *(see EcoStruxure Machine Expert, SystemConfigurationItf, Library Guide)*	Input for the axis that is to be controlled.
i_sFilename	STRING	File name of the binary motor data file that was created by using FC_MotorDataFileCreate.
i_etStorageLocation	ET_StorageLocation *(see page 31)*	Storage location to which the motor data is to be written. Default value: drive. The MotorIdentification *(see Lexium LXM62 Drive, Device Objects and Parameters,)* parameter of the axis has to be set according to the selected storage location. ● ET_StorageLocation.Drive => Motor without nameplate / 2 ● ET_StorageLocation.Encoder => Motor with nameplate / 0

Output	Data type	Description
q_xActive	BOOL	TRUE: The POU is active and has to be executed further. FALSE: The POU is inactive.
q_xReady	BOOL	TRUE: The POU is ready to operate and can accept user commands. FALSE: The function block is not ready to accept user commands.
q_xBusy	BOOL	TRUE: The POU executes the issued user command. FALSE: The POU is waiting for further user commands.
q_xDone	BOOL	TRUE: The user command has been executed. FALSE: The user command is being executed, or none has been issued yet.
q_etDiag	GD.ET_Diag *(see EcoStruxure Machine Expert, PD_GlobalDiagnostics, Library Guide)*	General library-independent statement on the diagnostic. A value unequal to GD.ET_Diag.Ok corresponds to a diagnostic message.

Output	Data type	Description
q_etDiagExt	ET_DiagExt *(see page 26)*	POU-specific output on the diagnostic. q_etDiag = GD.ET_Diag.Ok -> status message q_etDiag <> GD.ET_Diag.Ok -> diagnostic message
q_sMsg	STRING[80]	Event-triggered message which gives more detailed information on the diagnostic state.

Diagnostic Messages

q_etDiag	q_etDiagExt	Enumeration value	Description
OK	CheckingMotorDataInDrive *(see page 60)*	6	The POU verifies whether the drive contains motor data.
OK	Disabled *(see page 60)*	16	The POU is disabled.
OK	Done *(see page 60)*	10	The command has successfully been performed.
OK	Prepare *(see page 62)*	11	The POU is preparing the execution.
OK	WaitForExecute *(see page 63)*	3	The POU is waiting for execution.
OK	WaitForSercosPhase2 *(see page 63)*	2	The POU is waiting for Sercos phase 2.
DriveConditionInvalid	MotorDataAlreadyStoredInDrive *(see page 61)*	12	Motor data is already contained in the drive.
FileHandlingInvalid	FileAccessNotPossible *(see page 60)*	14	The file cannot be accessed.
FileHandlingInvalid	MotorDataFileNotFound *(see page 62)*	13	File not found.
InputParameterInvalid	FunctionNotSupportedByThisDevice *(see page 61)*	4	The drive does not support the function.
InputParameterInvalid	InvalidStorageLocation *(see page 61)*	41	The selected storage location for the electronic motor nameplate is invalid.
SercosConditionInvalid	SercosCommunicationNotPossible *(see page 62)*	7	Sercos communication is not possible.
UnexpectedProgramBehavior	ReinitializationFailed *(see page 62)*	15	Reinitialization of the drive was not successful.

CheckingMotorDataInDrive

Enumeration name:	CheckingMotorDataInDrive
Enumeration value:	6
Description:	The POU verifies whether the drive contains motor data.

The function block is being executed. Waiting until q_xDone has the value TRUE.

Disabled

Enumeration name:	Disabled
Enumeration value:	16
Description:	The POU is disabled.

Cause	Solution
The POU is disabled.	Set the i_xEnable input from FALSE to TRUE to enable the POUs.

Done

Enumeration name:	Done
Enumeration value:	10
Description:	The command has successfully been performed.

The motor data has been written successfully.

FileAccessNotPossible

Enumeration name:	FileAccessNotPossible
Enumeration value:	14
Description:	The file cannot be accessed.

Cause	Solution
The file cannot be accessed.	Verify whether the file is used elsewhere in the project.

FunctionNotSupportedByThisDevice

Enumeration name:	FunctionNotSupportedByThisDevice
Enumeration value:	4
Description:	The drive does not support the function.

Cause	Solution
Invalid object type	Verify the selected object.

InvalidStorageLocation

Enumeration name:	InvalidStorageLocation
Enumeration value:	41
Description:	The selected storage location for the electronic motor nameplate is invalid.

Cause	Solution
There is no Hiperface encoder connected to the physical machine encoder input.	Connect a Hiperface encoder to the physical machine encoder input.
The encoder does not provide sufficient memory or there is no encoder.	Connect an encoder with a Hiperface interface with a minimum of 2 kBytes physical memory.

MotorDataAlreadyStoredInDrive

Enumeration name:	MotorDataAlreadyStoredInDrive
Enumeration value:	12
Description:	Motor data is already contained in the drive.

Cause	Solution
The drive already contains the motor data.	The data must be deleted before new data can be written. Motor data can be deleted using FB_MotorDataDelete.

MotorDataFileNotFound

Enumeration name:	MotorDataFileNotFound
Enumeration value:	13
Description:	File not found.

Cause	Solution
Motor data file not found.	Verify the file name.

Prepare

Enumeration name:	Prepare
Enumeration value:	11
Description:	The POU is preparing the execution.

The POU in the q_xBusy execution is TRUE.

ReinitializationFailed

Enumeration name:	ReinitializationFailed
Enumeration value:	15
Description:	Reinitialization of the drive was not successful.

Cause	Solution
Reinitialization was not successful.	Verify the Sercos phase.

SercosCommunicationNotPossible

Enumeration name:	SercosCommunicationNotPossible
Enumeration value:	7
Description:	Sercos communication is not possible.

Cause	Solution
Sercos communication is not possible.	Verify the Sercos phase and the wiring.

WaitForExecute

Enumeration name:	WaitForExecute
Enumeration value:	3
Description:	The POU is waiting for execution.

The POU is active and ready for execution. q_xReady is TRUE.

WaitForSercosPhase2

Enumeration name:	WaitForSercosPhase2
Enumeration value:	2
Description:	The POU is waiting for Sercos phase 2.

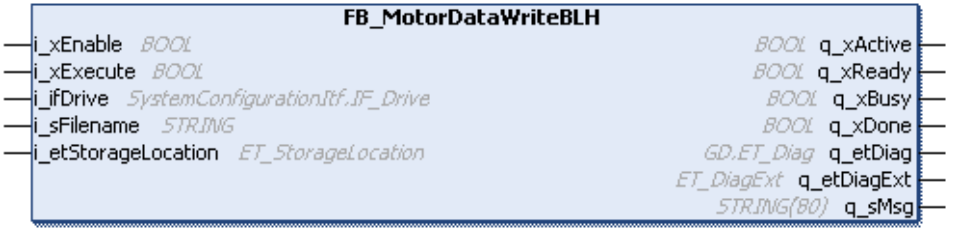
The POU is active and is waiting for Sercos phase 2. q_xReady is FALSE.

Section 3.5
FB_MotorDataWriteBLH

FB_MotorDataWriteBLH - General Information

Overview

Type:	Function block
Available as of:	V1.1.1.5
Inherits from:	–
Implements:	–



Task

This function block writes the motor data to the drive. Also refer to the chapter Asynchronous Motor *(see EcoStruxure Machine Expert, 3rd Party Motor, Library Guide)*.

NOTE: Execute this POU only when required for application needs. The encoder memory area has a limited number of write cycles, frequent execution of the POU can lead to encoder damage.

NOTE: Incorrect machine encoder typeplate data has a direct influence on the behavior of the controller. This function may only be used by technically qualified personnel.

⚠ DANGER

UNINTENDED BEHAVIOR OF THE MOTOR

- Verify assignment of machine encoder data to the motor before writing.
- Ensure that no one is in the operating zone during the startup.

Failure to follow these instructions will result in death or serious injury.

NOTE: A deactivated monitoring function can lead to a possible undetected overheating of the motor.

⚠ DANGER

FIRE DUE TO OVERHEATING OF THE MOTOR

- Verify assignment of motor data to the motor before writing.
- Use appropriate methods to verify thermal overload of the motor.
- Ensure that no one is in the operating zone during the startup.

Failure to follow these instructions will result in death or serious injury.

Requirements

- Sercos must be in phase 2.
- No motor data must be stored in the selected drive (use `FB_MotorDataDelete`, if necessary).
- The `MotorIdentification` *(see Lexium LXM62 Drive, Device Objects and Parameters,)* parameter of the axis must be set to the correct value.

Description

This POU receives a BLH file with motor data and writes it to the selected axis.

The motor data is stored to the selected storage location.

- In the memory of the encoder, if sufficient memory is available in the encoder.
- Stored in the servo amplifier if the encoder has no or insufficient memory or no encoder is used.

The motor data is permanently stored. Thus, the motor data must be written only once.

NOTE: When replacing drives or motors, make sure that the correct motor data is stored in the drive or motor. Under certain circumstances, the motor data must be rewritten.

If motor data is to be modified (for example, because the motor was replaced), you must first delete the existing motor data with `FB_MotorDataDelete` *(see page 45)*. Motor data must not be overwritten.

NOTE: The POU has a long processing time (> 1 s). Thus, it must be executed in a task that is not time-critical and with an appropriate watchdog time.

Interface

Input	Data type	Description
i_xEnable	BOOL	A rising edge FALSE -> TRUE activates the POU, a falling edge TRUE -> FALSE deactivates the POU. A deactivated POU does not execute any actions.
i_xExecute	BOOL	FALSE -> TRUE: The function block writes the motor data into the selected axis. During the writing procedure q_xBusy = TRUE. As soon as the writing procedure is completed q_xDone changes to TRUE.
i_ifDrive	SystemConfigurationItf.IF_Drive *(see EcoStruxure Machine Expert, SystemConfigurationItf, Library Guide)*	Input for the axis that is to be controlled.
i_sFilename	STRING	The name of the BLH file
i_etStorageLocation	ET_StorageLocation *(see page 31)*	Storage location to which the motor data is to be written. Default value: drive. The MotorIdentification *(see Lexium LXM62 Drive, Device Objects and Parameters,)* parameter of the axis has to be set according to the selected storage location. ● ET_StorageLocation.Drive => Motor without nameplate / 2 ● ET_StorageLocation.Encoder => Motor with nameplate / 0

Output	Data type	Description
q_xActive	BOOL	TRUE: The POU is active and has to be executed further. FALSE: The POU is inactive.
q_xReady	BOOL	TRUE: The POU is ready to operate and can accept user commands. FALSE: The function block is not ready to accept user commands.
q_xBusy	BOOL	TRUE: The POU executes the issued user command. FALSE: The POU is waiting for further user commands.
q_xDone	BOOL	TRUE: The user command has been executed. FALSE: The user command is being executed, or none has been issued yet.

Output	Data type	Description
`q_etDiag`	GD.ET_Diag *(see EcoStruxure Machine Expert, PD_GlobalDiagnostics, Library Guide)*	General library-independent statement on the diagnostic. A value unequal to `GD.ET_Diag.Ok` corresponds to a diagnostic message.
`q_etDiagExt`	ET_DiagExt *(see page 26)*	POU-specific output on the diagnostic. `q_etDiag = GD.ET_Diag.Ok` -> status message `q_etDiag <> GD.ET_Diag.Ok` -> diagnostic message
`q_sMsg`	STRING[80]	Event-triggered message which gives more detailed information on the diagnostic state.

Diagnostic Messages

q_etDiag	q_etDiagExt	Enumeration value	Description
`OK`	`CheckingMotorDataInDrive` *(see page 68)*	6	The POU verifies whether the drive contains motor data.
`OK`	`Disabled` *(see page 68)*	16	The POU is disabled.
`OK`	`Done` *(see page 68)*	10	The command has successfully been performed.
`OK`	`Prepare` *(see page 70)*	11	The POU is preparing the execution.
`OK`	`WaitForExecute` *(see page 70)*	3	The POU is waiting for execution.
`OK`	`WaitForSercosPhase2` *(see page 71)*	2	The POU is waiting for Sercos phase 2.
`DriveConditionInvalid`	`MotorDataAlreadyStoredIn Drive` *(see page 69)*	12	Motor data is already contained in the drive.
`FileHandlingInvalid`	`FileAccessNotPossible` *(see page 68)*	14	The file cannot be accessed.
`InputParameterInvalid`	`FunctionNotSupportedBy ThisDevice` *(see page 69)*	4	The drive does not support the function.
`InputParameterInvalid`	`InvalidStorageLocation` *(see page 69)*	41	The selected storage location for the electronic motor nameplate is invalid.
`SercosConditionInvalid`	`SercosCommunicationNot Possible` *(see page 70)*	7	Sercos communication is not possible.
`UnexpectedProgram Behavior`	`ReinitializationFailed` *(see page 70)*	15	Reinitialization of the drive was not successful.

CheckingMotorDataInDrive

Enumeration name:	CheckingMotorDataInDrive
Enumeration value:	6
Description:	The POU verifies whether the drive contains motor data.

The function block is being executed. Waiting until q_xDone has the value TRUE.

Disabled

Enumeration name:	Disabled
Enumeration value:	16
Description:	The POU is disabled.

Cause	Solution
The POU is disabled.	Set the i_xEnable input from FALSE to TRUE to enable the POUs.

Done

Enumeration name:	Done
Enumeration value:	10
Description:	The command has successfully been performed.

The motor data has been written successfully.

FileAccessNotPossible

Enumeration name:	FileAccessNotPossible
Enumeration value:	14
Description:	The file cannot be accessed.

Cause	Solution
The file cannot be accessed.	Verify whether the file is used elsewhere in the project.

FunctionNotSupportedByThisDevice

Enumeration name:	FunctionNotSupportedByThisDevice
Enumeration value:	4
Description:	The drive does not support the function.

Cause	Solution
Invalid object type	Verify the selected object.

InvalidStorageLocation

Enumeration name:	InvalidStorageLocation
Enumeration value:	41
Description:	The selected storage location for the electronic motor nameplate is invalid.

Cause	Solution
There is no Hiperface encoder connected to the physical machine encoder input.	Connect a Hiperface encoder to the physical machine encoder input.
The encoder does not provide sufficient memory or there is no encoder.	Connect an encoder with a Hiperface interface with a minimum of 2 kBytes physical memory.

MotorDataAlreadyStoredInDrive

Enumeration name:	MotorDataAlreadyStoredInDrive
Enumeration value:	12
Description:	Motor data is already contained in the drive.

Cause	Solution
The drive already contains the motor data.	The existing data must be deleted before new data can be written. Motor data can be deleted using FB_MotorDataDelete.

Prepare

Enumeration name:	Prepare
Enumeration value:	11
Description:	The POU is preparing the execution.

The POU in the q_xBusy execution is TRUE.

ReinitializationFailed

Enumeration name:	ReinitializationFailed
Enumeration value:	15
Description:	Reinitialization of the drive was not successful.

Cause	Solution
Reinitialization was not successful.	Verify the Sercos phase.

SercosCommunicationNotPossible

Enumeration name:	SercosCommunicationNotPossible
Enumeration value:	7
Description:	Sercos communication is not possible.

Cause	Solution
Sercos communication is not possible.	Verify the Sercos phase and the wiring.

WaitForExecute

Enumeration name:	WaitForExecute
Enumeration value:	3
Description:	The POU is waiting for execution.

The POU is active and ready for execution. q_xReady is TRUE.

WaitForSercosPhase2

Enumeration name:	WaitForSercosPhase2
Enumeration value:	2
Description:	The POU is waiting for Sercos phase 2.

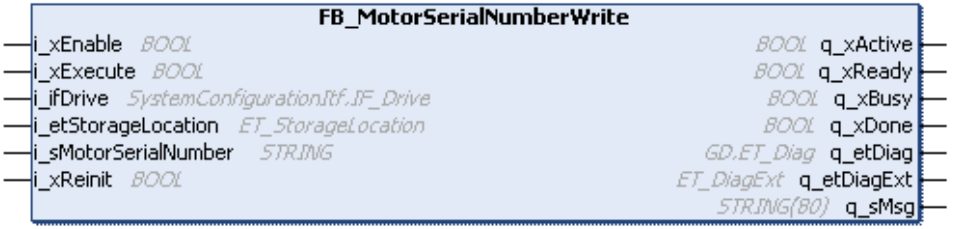
The POU is active and is waiting for Sercos phase 2. q_xReady is FALSE.

Section 3.6
FB_MotorSerialNumberWrite

FB_MotorSerialNumberWrite - General Information

Overview

Type:	Function block
Available as of:	V1.1.1.5
Inherits from:	–
Implemented:	–



Task

Writing the serial number of a motor.

Description

This function block writes the motor serial number in the electronic motor nameplate.

Interface

Input	Data type	Description
i_xEnable	BOOL	A rising edge FALSE -> TRUE activates the POU, a falling edge TRUE -> FALSE deactivates the POU. A deactivated POU does not execute any actions.
i_xExecute	BOOL	FALSE -> TRUE: The POU writes the motor data serial number to the selected axis. During the writing procedure q_xBusy = TRUE. As soon as the writing procedure is completed q_xDone changes to TRUE.
i_ifDrive	SystemConfigurationItf.IF_Drive *(see EcoStruxure Machine Expert, SystemConfigurationItf, Library Guide)*	Input for the axis that is to be controlled.
i_etStorageLocation	ET_StorageLocation *(see page 31)*	Storage location to which the serial number is to be written (default value: drive).
i_sMotorSerialNumber	STRING	Serial number of the motor, maximum of 20 characters
i_xReinit	BOOL	TRUE: After writing the motor serial number, the drive is reinitialized and the motor nameplate is reread. FALSE: The drive is not reinitialized and the motor nameplate is not reread.

Output	Data type	Description
q_xActive	BOOL	TRUE: The POU is active and has to be executed further. FALSE: The POU is inactive.
q_xReady	BOOL	TRUE: The POU is ready to operate and can accept user commands. FALSE: The function block is not ready to accept user commands.
q_xBusy	BOOL	TRUE: The POU executes the issued user command. FALSE: The POU is waiting for further user commands.
q_xDone	BOOL	TRUE: The user command has been executed. FALSE: The user command is being executed, or none has been issued yet.
q_etDiag	GD.ET_Diag *(see EcoStruxure Machine Expert, PD_GlobalDiagnostics, Library Guide)*	General, library-independent statement on the diagnostic. A value unequal to GD.ET_Diag.Ok corresponds to a diagnostic message.

Output	Data type	Description
q_etDiagExt	ET_DiagExt *(see page 26)*	POU-specific output on the diagnostic. q_etDiag = GD.ET_Diag.Ok -> status message q_etDiag <> GD.ET_Diag.Ok -> diagnostic message
q_sMsg	STRING[80]	Event-triggered message which gives more detailed information on the diagnostic state.

Diagnostic Messages

q_etDiag	q_etDiagExt	Enumeration value	Description
OK	Disabled *(see page 74)*	16	The POU is disabled.
OK	Done *(see page 75)*	10	The command has successfully been performed.
OK	Executing *(see page 75)*	46	The POU is being executed.
OK	WaitForExecute *(see page 76)*	3	The POU is waiting for execution.
OK	WaitForSercosPhase2 *(see page 76)*	2	The POU is waiting for Sercos phase 2.
DriveConditionInvalid	OperationNotAllowed *(see page 76)*	45	This operation is not permitted for the selected axis.
InputParameterInvalid	FunctionNotSupportedBy ThisDevice *(see page 75)*	4	The drive does not support the function.
InputParameterInvalid	InputStringTooLarge *(see page 75)*	18	The entered strings are too long.
SercosConditionInvalid	SercosCommunicationNot Possible *(see page 76)*	7	Sercos communication is not possible.

Disabled

Enumeration name:	Disabled
Enumeration value:	16
Description:	The POU is disabled.

Cause	Solution
The POU is disabled.	Set the i_xEnable input from FALSE to TRUE to enable the POUs.

Done

Enumeration name:	Done
Enumeration value:	10
Description:	The command has successfully been performed.

The motor serial number was successfully written.

Executing

Enumeration name:	Executing
Enumeration value:	46
Description:	The POU is being executed.

The POU is being executed.

FunctionNotSupportedByThisDevice

Enumeration name:	FunctionNotSupportedByThisDevice
Enumeration value:	4
Description:	The drive does not support the function.

Cause	Solution
Invalid object type	Verify the selected object.

InputStringTooLarge

Enumeration name:	InputStringTooLarge
Enumeration value:	18
Description:	The entered strings are too long.

Cause	Solution
The specified motor serial number is longer than 20 characters.	Specify a motor serial number with fewer than 20 characters.

OperationNotAllowed

Enumeration name:	OperationNotAllowed
Enumeration value:	45
Description:	This operation is not permitted for the selected axis.

Cause	Solution
The operation is not permitted for the selected axis, for example because a Schneider Electric servo motor is used.	Select an axis that is permitted for this operation.

SercosCommunicationNotPossible

Enumeration name:	SercosCommunicationNotPossible
Enumeration value:	7
Description:	Sercos communication is not possible.

Cause	Solution
Sercos communication is not possible.	Verify the Sercos phase and the wiring.

WaitForExecute

Enumeration name:	WaitForExecute
Enumeration value:	3
Description:	The POU is waiting for execution.

The POU is active and ready for execution. q_xReady is TRUE.

WaitForSercosPhase2

Enumeration name:	WaitForSercosPhase2
Enumeration value:	2
Description:	The POU is waiting for Sercos phase 2.

The POU is active and is waiting for Sercos phase 2. q_xReady is FALSE.

Chapter 4
Functions

What Is in This Chapter?

This chapter contains the following sections:

Section 4.1
FC_EtDiagExtToString

FC_EtDiagExtToString - General Information

Overview

Type:	Function
Available as of:	V1.0.0.0

Task

Converting an enumeration element of type ET_DiagExt *(see page 26)* of the UserMotorTypePlate library to a string.

Description

Every enumeration element has a name or value. The return value of the function is the name of the enumeration element.

Interface

Input	Data type	Description
i_etDiagExt	ET_DiagExt *(see page 26)*	The ET_DiagExt value to be converted.

Output	Data type	Description
q_etDiag	GD.ET_Diag *(see EcoStruxure Machine Expert, PD_GlobalDiagnostics, Library Guide)*	General, library-independent statement on the diagnostic. A value not equal to ET_Diag.Ok corresponds to a diagnostic message.
q_etDiagExt	ET_DiagExt *(see page 26)*	POU-specific output on the diagnostic. q_etDiag = ET_Diag.Ok -> Status message q_etDiag <> ET_Diag.Ok -> diagnostic message

Return Value

Data type	Description
STRING[80]	Name of the enumeration element that was transferred at the input i_etDiagExt.

Section 4.2
FC_MotorDataFileCreate

FC_MotorDataFileCreate - General Information

Overview

Type:	Function
Available as of:	V1.0.0.0
Inherits from:	–
Implements:	–

Task

Creating a binary motor data file. Also refer to the chapter Asynchronous Motor *(see EcoStruxure Machine Expert, 3rd Party Motor, Library Guide)*.

NOTE: Incorrect machine encoder typeplate data has a direct influence on the behavior of the controller. This function may only be used by technically qualified personnel.

⚠ DANGER

UNINTENDED BEHAVIOR OF THE MOTOR

- The technical data of the motor manufacturer must be observed and respected.
- The motor data of the parameterized variables must concur with the motor data of the manufacturer.
- If the definition of the motor data does not correspond to the motor, you must convert it to the required motor data.
- Ensure that no one is in the operating zone during the startup.

Failure to follow these instructions will result in death or serious injury.

NOTE: A deactivated monitoring function can lead to a possible undetected overheating of the motor.

⚠ DANGER

FIRE DUE TO OVERHEATING OF THE MOTOR

- The technical data of the motor manufacturer must be observed and respected.
- The motor data of the parameterized variables must concur with the motor data of the manufacturer.
- If the definition of the motor data does not correspond to the motor, you must convert it to the required motor data.
- Use appropriate methods to verify thermal overload of the motor.
- Ensure that no one is in the operating zone during the startup.

Failure to follow these instructions will result in death or serious injury.

Description

The function creates a binary motor data file that can be used in combination with `FB_MotorDataWrite` *(see page 56)* to write the motor data into an axis. You must specify the name of the file and a structure that contains the motor data.

NOTE: The POU has a long processing time (> 1 s). Thus, it must be executed in a task that is not time-critical and with an appropriate watchdog time.

Interface

Input	Data type	Description
`i_sFilename`	STRING[80]	Name of the file that is created.
`i_stUserMotorData`	ST_UserMotorData *(see page 98)*	Structure that contains the application data that is written into the file.

Output	Data type	Description
`q_etDiag`	GD.ET_Diag *(see EcoStruxure Machine Expert, PD_GlobalDiagnostics, Library Guide)*	General library-independent statement on the diagnostic. A value unequal to `GD.ET_Diag.Ok` corresponds to a diagnostic message.
`q_etDiagExt`	ET_DiagExt *(see page 26)*	POU-specific output on the diagnostic. `q_etDiag = GD.ET_Diag.Ok` -> status message `q_etDiag <> GD.ET_Diag.Ok` -> diagnostic message
`q_sMsg`	STRING[80]	Event-triggered message which gives more detailed information on the diagnostic state.

Return Value

Data type	Description
BOOL	TRUE if file creation was successful.

Diagnostic Messages

q_etDiag	q_etDiagExt	Enumeration value	Description
OK	Disabled *(see page 84)*	16	The POU is disabled.
FileHandlingInvalid	CouldNotCreateFile *(see page 83)*	40	The file for the motor data could not be created.
InputParameterInvalid	BrakeInvalid *(see page 82)*	37	The uiBrake value is invalid.
InputParameterInvalid	ContinuousStallCurrentInvalid *(see page 83)*	30	The value for the standstill rated current is invalid.
InputParameterInvalid	ContinuousStallTorqueInvalid *(see page 83)*	31	The value for the standstill torque is invalid.
InputParameterInvalid	InputStringTooLarge *(see page 84)*	18	The entered strings are too long.
InputParameterInvalid	InvalidEncoderType *(see page 84)*	27	The specified encoder type is invalid.
InputParameterInvalid	InvalidMotorName *(see page 85)*	26	The motor name is invalid.
InputParameterInvalid	InvalidNumberOfPolePairs *(see page 85)*	23	The specified pole pair number is invalid.
InputParameterInvalid	MandatoryParameterInvalid *(see page 85)*	38	The mandatory parameter is invalid.
InputParameterInvalid	MaxSpeedInvalid *(see page 86)*	35	The maximum speed of rotation is invalid.
InputParameterInvalid	MotorCosPhiInvalid *(see page 86)*	24	Cosine phi of the motor is invalid.
InputParameterInvalid	MotorInertiaInvalid *(see page 86)*	36	The mass inertia of the motor is invalid.
InputParameterInvalid	MotorTypeNotSupported *(see page 87)*	39	The motor type is not supported.
InputParameterInvalid	NominalCurrentInvalid *(see page 87)*	22	The rated current is invalid.

q_etDiag	q_etDiagExt	Enumeration value	Description
InputParameterInvalid	NominalFrequencyInvalid *(see page 87)*	20	The rated frequency is invalid.
InputParameterInvalid	NominalPowerInvalid	49	The nominal power is invalid.
InputParameterInvalid	NominalSpeedInvalid *(see page 88)*	19	The rated velocity is invalid.
InputParameterInvalid	NominalVoltageInvalid *(see page 88)*	21	The rated voltage is invalid.
InputParameterInvalid	PeakCurrentInvalid *(see page 89)*	28	The peak current of the motor is invalid.
InputParameterInvalid	PeakTorqueInvalid *(see page 89)*	32	The peak torque of the motor is invalid.
InputParameterInvalid	PhaseResistanceInvalid *(see page 89)*	33	The winding resistance is invalid.
InputParameterInvalid	QuadraturePhaseInductance Invalid *(see page 90)*	34	The inductance value is invalid.
InputParameterInvalid	RotatingFieldDirection Invalid *(see page 90)*	25	The rotary field direction is invalid.
InputParameterInvalid	SingleturnResolution Invalid *(see page 90)*	42	The specified singleturn resolution of the encoder is invalid.

BrakeInvalid

Enumeration name:	BrakeInvalid
Enumeration value:	37
Description:	The uiBrake value is invalid.

Cause	Solution
uiBrake has an invalid value. Only the values 0 and 1 are permitted.	Set uiBrake to a valid value.

ContinuousStallCurrentInvalid

Enumeration name:	ContinuousStallCurrentInvalid
Enumeration value:	30
Description:	The value for the standstill rated current is invalid.

Cause	Solution
The standstill rated current must be > 0.	Adapt the value for the standstill rated current.

ContinuousStallTorqueInvalid

Enumeration name:	ContinuousStallTorqueInvalid
Enumeration value:	31
Description:	The value for the standstill torque is invalid.

Cause	Solution
The standstill torque must be > 0.	Adapt the value for the standstill torque.

CouldNotCreateFile

Enumeration name:	CouldNotCreateFile
Enumeration value:	40
Description:	The file for the motor data could not be created.

Cause	Solution
There is already a file with the same name.	Use a different file name.
Not enough memory available on the flash disk.	Clear disk space or use a flash disk with more memory.
Flash disk defective.	Replace the flash disk.

Disabled

Enumeration name:	Disabled
Enumeration value:	16
Description:	The POU is disabled.

Cause	Solution
The POU is disabled.	Set the i_xEnable input from FALSE to TRUE to enable the POUs.

InputStringTooLarge

Enumeration name:	InputStringTooLarge
Enumeration value:	18
Description:	The entered strings are too long.

Cause	Solution
The parameters sMotorname, sMotorArticleNumber, or sMotorSerialnumber have an invalid length. A maximum of 20 characters is permitted.	Adapt the string length.

InvalidEncoderType

Enumeration name:	InvalidEncoderType
Enumeration value:	27
Description:	The specified encoder type is invalid.

Cause	Solution
uiEncoderType has an invalid value.	Adapt the value.

InvalidMotorName

Enumeration name:	InvalidMotorName
Enumeration value:	26
Description:	The motor name is invalid.

Cause	Solution
The motor names SH, BSH, or BMH are not permitted.	Do not use an inappropiate motor name.

InvalidNumberOfPolePairs

Enumeration name:	InvalidNumberOfPolePairs
Enumeration value:	23
Description:	The specified pole pair number is invalid.

Cause	Solution
The pole pair number must not be 0.	Enter the pole pair number of the motor.

MandatoryParameterInvalid

Enumeration name:	MandatoryParameterInvalid
Enumeration value:	38
Description:	The mandatory parameter is invalid.

Cause	Solution
A required parameter has an invalid value.	Adapt the parameter values.

MaxSpeedInvalid

Enumeration name:	MaxSpeedInvalid
Enumeration value:	35
Description:	The maximum speed of rotation is invalid.

Cause	Solution
The value 0 is not permitted.	Adapt the value.

MotorCosPhiInvalid

Enumeration name:	MotorCosPhiInvalid
Enumeration value:	24
Description:	Cosine phi of the motor is invalid.

Cause	Solution
The cosine phi of the motor must be greater than 0 and less than 1.	Adapt the value of cosine phi.

MotorInertiaInvalid

Enumeration name:	MotorInertiaInvalid
Enumeration value:	36
Description:	The mass inertia of the motor is invalid.

Cause	Solution
The mass inertia of the motor must not be 0.	Adapt the value for the mass inertia.

MotorTypeNotSupported

Enumeration name:	MotorTypeNotSupported
Enumeration value:	39
Description:	The motor type is not supported.

The following motor types are currently supported in the library:
- Rotary asynchronous motors
- Linear synchronous motors
- Rotary synchronous motors with six or more pole pairs

NominalCurrentInvalid

Enumeration name:	NominalCurrentInvalid
Enumeration value:	22
Description:	The rated current is invalid.

Cause	Solution
The rated current must be greater than 0.	Adapt the rated current.

NominalFrequencyInvalid

Enumeration name:	NominalFrequencyInvalid
Enumeration value:	20
Description:	The rated frequency is invalid.

Cause	Solution
The rated frequency must not be equal to 0.	Adapt the value of the rated frequency.

NominalPowerInvalid

Enumeration name:	NominalPowerInvalid
Enumeration value:	49
Description:	The nominal power is invalid.

Cause	Solution
The parameter i_stUserMotorData.stMotorDataACIM.rNominalPower is out of range.	i_stUserMotorData.stMotorDataACIM.rNominalPower must be in range of 0...110, and if i_stUserMotorData.stMotorDataACIM.rNominalPower = 0, then the value for i_stUserMotorData.stMotorDataACIM.uiNominalPower must be > 0. Adapt the values as described.

NominalSpeedInvalid

Enumeration name:	NominalSpeedInvalid
Enumeration value:	19
Description:	The rated velocity is invalid.

Cause	Solution
The rated velocity must not be equal to 0.	Adapt the value of the rated velocity.

NominalVoltageInvalid

Enumeration name:	NominalVoltageInvalid
Enumeration value:	21
Description:	The rated voltage is invalid.

Cause	Solution
The rated voltage must not be equal to or smaller than 0.	Adapt the value of the rated voltage.

PeakCurrentInvalid

Enumeration name:	PeakCurrentInvalid
Enumeration value:	28
Description:	The peak current of the motor is invalid.

Cause	Solution
The peak current must be greater than 0.	Adapt the peak current.

PeakTorqueInvalid

Enumeration name:	PeakTorqueInvalid
Enumeration value:	32
Description:	The peak torque of the motor is invalid.

Cause	Solution
The peak torque must be greater than 0.	Adapt the value for the peak torque.

PhaseResistanceInvalid

Enumeration name:	PhaseResistanceInvalid
Enumeration value:	33
Description:	The winding resistance is invalid.

Cause	Solution
The winding resistance must be greater than 0.	Adapt the winding resistance.

QuadraturePhaseInductanceInvalid

Enumeration name:	QuadraturePhaseInductanceInvalid
Enumeration value:	34
Description:	The inductance value is invalid.

Cause	Solution
The inductance must be greater than 0.	Adapt the value.

RotatingFieldDirectionInvalid

Enumeration name:	RotatingFieldDirectionInvalid
Enumeration value:	25
Description:	The rotary field direction is invalid.

Cause	Solution
The rotary field direction must be 0 or 1.	Adapt the value of the rotary field direction.

SingleturnResolutionInvalid

Enumeration name:	SingleturnResolutionInvalid
Enumeration value:	42
Description:	The specified singleturn resolution of the encoder is invalid.

Cause	Solution
The singleturn data length and the singleturn resolution do not correspond.	Adapt singleturn data length. Adapt singleturn resolution.

Section 4.3
FC_MotorDataFileRead

FC_MotorDataFileRead - General Information

Overview

Type:	Function
Available as of:	V1.1.0.0
Inherits from:	–
Implemented:	–

Task

Converting a binary file with motor data into a data structure.

Description

This function reads the specified binary file and writes the motor data from the file to a `ST_UserMotorData` structure.

Interface

Input	Data type	Description
`i_sFilename`	STRING[80]	File name of the binary file with the motor data to be read.

Output	Data type	Description
`q_etDiag`	GD.ET_Diag *(see EcoStruxure Machine Expert, PD_GlobalDiagnostics, Library Guide)*	General, library-independent statement on the diagnostic. A value unequal to `GD.ET_Diag.Ok` corresponds to a diagnostic message.
`q_etDiagExt`	ET_DiagExt *(see page 26)*	POU-specific output on the diagnostic. `q_etDiag = GD.ET_Diag.Ok` -> status message `q_etDiag <> GD.ET_Diag.Ok` -> diagnostic message
`q_sMsg`	STRING[80]	Event-triggered message which gives more detailed information on the diagnostic state.

Input/Output	Data type	Description
`iq_stUserMotorData`	ST_UserMotorData	The motor data read from the `i_sFilename` file is written to this structure.

Return Value

Data type	Description
BOOL	TRUE = File was successfully read and the motor data is written to the `iq_stUserMotorData` structure. FALSE = File could not be read of an error occurred when writing the motor data (for further information see diagnostic outputs).

Diagnostic Messages

q_etDiag	q_etDiagExt	Enumeration value	Description
`OK`	`Disabled` *(see page 92)*	16	The POU is disabled.
`FileHandlingInvalid`	`FileAccessNotPossible` *(see page 93)*	14	The file cannot be accessed.
`FileHandlingInvalid`	`MotorDataFileNotFound` *(see page 93)*	13	File not found.

`Disabled`

Enumeration name:	`Disabled`
Enumeration value:	16
Description:	The POU is disabled.

Set the `i_xEnable` input from FALSE to TRUE to enable the POUs.

FileAccessNotPossible

Enumeration name:	FileAccessNotPossible
Enumeration value:	14
Description:	The file cannot be accessed.

Cause	Solution
The file is currently being used by another function block.	Only access the file if the file is not being used by another function block.

MotorDataFileNotFound

Enumeration name:	MotorDataFileNotFound
Enumeration value:	13
Description:	File not found.

Cause	Solution
The indicated file name is incorrect.	Verify i_sFilename parameter.

Chapter 5
Structures

What Is in This Chapter?

This chapter contains the following sections:

Section 5.1
ST_UserMachineEncoderData

ST_UserMachineEncoderData - General Description

Overview

Type:	Data structure
Available as of:	V1.0.0.0
Inherits from:	-

Description

This structure contains general machine encoder data for the machine encoder.

Structure Elements

Variable	Data type	Description
etEncoderType	ET_EncoderType *(see page 29)*	Type of encoder, for example SinCosHiperface, SinCos. Required parameter
uiEncoderMaxSpeed	UINT	Maximum speed of encoder [rpm]. Not required Default value: 0 If uiEncoderMaxSpeed is equal to 0, the speed of the encoder is not monitored. If uiEncoderMaxSpeed is greater than 0, the speed of the encoder is monitored.
uiEncoderMaxTemp	UINT	Maximum temperature of encoder [°C]. Not required Default value: 0 If uiEncoderTempSensor is equal to 0, the value is ignored. The maximum temperature of the encoder is not monitored. If uiEncoderTempSensor is 1 AND uiEncoderMaxTemp is not equal to 0, the temperature of the encoder is monitored according to the temperature level.

Variable	Data type	Description
`uiEncoderTempSensor`	UINT (0...1)	Indicates whether the encoder has an integrated temperature sensor. Not required 0: no temperature sensor; if a sensor exists, the encoder temperature monitoring is disabled 1: temperature sensor; encoder temperature is monitored according to `uiEncoderMaxTemp` when `uiEncoderMaxTemp` is not equal to 0
`uiEncoderNumberOfTurns`	UINT	Number of the displayable revolutions of the encoder. Not required For Hiperface encoders, the value is ignored, the following values are considered instead: ● For encoders with generic typeplate: Values are retrieved from encoder generic typeplate. ● For encoders without generic typeplate: SC• 60, SR• 50, SK• 36 (•=S or M), SEK 37, SEL 37, TTK 70 Schneider Electric version, L 230: Hardcoded values are used.
`uiEncoderLinesPerRevolution`	UINT	Number of encoder periods per revolution. Not required For Hiperface encoders, the value is ignored, the following values are considered instead: ● For encoders with generic typeplate: Values are retrieved from encoder generic typeplate. ● For encoders without generic typeplate: SC• 60, SR• 50, SK• 36 (•=S or M), SEK 37, SEL 37, TTK 70 Schneider Electric version, L 230: Hardcoded values are used.

Section 5.2
ST_UserMotorData

ST_UserMotorData - General Information

Overview

Type:	Data structure
Available as of:	V1.0.0.0
Inherits from:	-

Description

This structure contains general motor data for all motor types.

Structure Elements

Variable	Data type	Description
etMotorType	ET_MotorType *(see page 30)*	Motor types: ● Rotary_PMSM ● Linear_PMSM ● Rotary_ACIM Required parameter
sMotorname	STRING[80]	Name of the motor (is displayed in the MotorType parameter), string with a maximum of 20 characters. Not required
sMotorSerialNumber	STRING[80]	Serial number of the motor (displayed in the SerialNumberMotor parameter). Not required
sMotorArticleNumber	STRING[80]	Item no. of the motor (displayed in the PartNumberMotor parameter). Not required
stMotorDataACIM	ST_UserMotorDataACIM *(see page 99)*	Motor data for asynchronous motors. This structure must be filled in if etMotorType is set to the value Rotary_ACIM.
stMotorDataPMSM	ST_UserMotorDataPMSM *(see page 104)*	Motor data for synchronous motors. This structure must be filled in if etMotorType is set to the value Rotary_PMSM or Linear_PMSM.

Section 5.3
ST_UserMotorDataACIM

ST_UserMotorDataACIM - General Information

Overview

Type:	Data structure
Available as of:	V1.0.0.0
Inherits from:	-

Description

This structure contains motor data, specifically for asynchronous motors (ACIM, AC induction motors).

Structure Elements

Variable	Data type	Description
uiNominalSpeed	UINT	Rated velocity in [rpm]. Required parameter
uiNominalFrequency	UINT	Rated frequency in [Hz]. Required parameter
rNominalPower	REAL	Rated power in [kW]. Required parameter
uiNominalPower	UINT	The parameter rNominalPower is prefered. The parameter uiNominalPower is used when rNominalPower is 0. Rated power in [W] Not required
rNominalVoltage	REAL	Rated voltage in [Vrms]. Required parameter
rNominalCurrent	REAL	Rated current in [Arms]. Required parameter
rMotorCosPhi	REAL	Cosine (phi) of the motor. Required parameter

Variable	Data type	Description
uiEncoderType	UINT	Encoder type (see ET_EncoderType *(see page 29)*) ● 0: no encoder ● 1: SinCos Hiperface encoder ● 2: SinCos encoder Required parameter
uiBrake	UINT	Indicates whether a brake is integrated in the motor: ● 0: no brake integrated ● 1: brake integrated Required parameter
uiTempSensorType	UINT	Type of the thermal sensor: ● 0: without function ● 1: nonlinear sensor (for example, PTC) ● 2: linear sensor (for example, KTY) Required parameter
rStatorResistance	REAL	Stator resistor of the motor (phase to phase) in [Ω]. Not required
rRotorResistance	REAL	Rotor resistor of the motor (phase to phase) in [Ω]. Not required
rMainInductance	REAL	Main inductance of the motor (phase to phase) in [µH] Not required
rStatorLeakageInductance	REAL	Stator leakage inductance of the motor (phase to phase) in [µH] Not required
rRotorLeakInductance	REAL	Rotor leakage inductance of the motor (phase to phase) in [µH] Not required
uiPolePair	UINT	Number of pairs of poles If uiPolePair is set to 0, the parameter is calculated by FC_MotorDataFileCreate. Not required
uiRotatingFieldDirection	UINT	Direction of the rotary field ● 0: counterclockwise ● 1: clockwise If the motor phases, ● U -> U ● V -> V ● W -> W are connected to the drive; then this parameter must be set to the value 1 (clockwise). Not required
uiMaxMotorTemperature	UINT	Maximum temperature of the motor in [°C] (default value: 130 °C). Not required

Variable	Data type	Description
`uiTempSensorResistance Overtemp`	UINT	Minimum resistance of the temperature sensor if the temperature is too high in [Ω] (default value: 4 kΩ). The parameter can only be used for PTC sensors. `FC_MotorDataFileCreate` calculates the sensor characteristics depending on this parameter. The value is used only if the type of the temperature sensor is 1. Not required
`auiTempSensorChracteristics`	ARRAY [0..19] OF UINT	Characteristics of the thermal sensor. This parameter is used only in the case of a linear sensor. In this case, you must specify the values: ● `auiTempSensorChracteristics [0]`: resistance in [Ω] for -30 °C ● `auiTempSensorChracteristics [1]`: resistance in [Ω] for -20 °C ● `auiTempSensorChracteristics [2]`: resistance in [Ω] for -10 °C ● ... ● `auiTempSensorChracteristics [19]`: resistance in [Ω] for 160 °C Not required
`uiMaxSpeed`	UINT	Maximum speed of rotation of the motor in [rpm]. Not required
`rPeakCurrent`	REAL	Peak current of the motor in [Arms]. If `rPeakCurrent` is equal to 0, the 1.5-fold value of the rated current is used. Not required
`uiEncoderMaxSpeed`	UINT	Maximum speed of rotation of the encoder [rpm]. Not required
`uiEncoderMaxTemp`	UINT	Maximum temperature of the encoder in [°C]. Not required
`uiEncoderTempSensor`	UINT	Indicates whether the encoder has an integrated temperature sensor. ● 0: no temperature sensor ● 1: temperature sensor Not required
`uiEncoderNumberOfTurns`	UINT	Number of the displayable revolutions of the encoder. Required parameter for SinCos encoder (without Hiperface interface)
`uiEncoderLinesPerRevolution`	UINT	Number of encoder periods per revolution. Required parameter for SinCos encoder (without Hiperface interface)

Variable	Data type	Description
wBrakeType	WORD	Specifies the type of the brake if present (default value: 0). 0: default holding brake 1: inverted holding brake Not required
uiBrakeDisconnectionTime	UINT	Period until the brake is switched off in [ms]. If uiBrakeDisconnectionTime is equal to 0, the drive uses the default value 100 ms. Not required
uiBrakeCouplingTime	UINT	Period until the brake is engaged in [ms]. If uiBrakeCouplingTime is equal to 0, the drive uses the default value 100 ms. Not required
rBrakeMinVoltage	REAL	Minimum control voltage that is required to use the brake in [V]. If rBrakeMinVoltage is equal to 0, the drive does not verify the brake voltage. Not required
rBrakeMaxVoltage	REAL	Maximum control voltage that is permitted to use the brake in [V]. If rBrakeMaxVoltage is equal to 0, the drive does not verify the brake voltage. Not required
uiBrakeNomCurrent	UINT	Rated current of the brake in [mA]. Not required
uiThermalConstant	UINT	Thermal constant for monitoring motor overheating in [ms]. The parameter defines for how long the motor can be operated at peak current without being overloaded. The default value is 1000 ms. Not required

Example

In the following example, an asynchronous motor is configured for the U/f operation mode. The motor has the following data:

- Rated speed = 1380 rpm
- Motor cosine phi = 0.7
- Rated frequency = 50 Hz
- Rated power = 370 W
- Rated voltage = 400 V
- Rated current = 1.14 A
- The start-up current is 3.5 times higher than the rated current (in this example the start-up current is used as peak current).
- The motor has no encoder.
- The motor has a nonlinear temperature sensor. If the motor temperature is too high, then the temperature sensor has a resistance > 4 kΩ.
- The motor has no brake.
- The motor phases U => U, V => V, W => W are connected to the drive (the direction of the electric rotating field is clockwise).

The motor data is integrated into the data structure as follows:

```
stUserMotorData.eMotorType := Rotary_ACIM;
stUserMotorData.sMotorname := 'SEW-Eurodrive';
stUserMotorData.sMotorArticleNumber := 'DRS71S4/FF/TH/AV1H';
stUserMotorData.sMotorSerialNumber := '01.1710743001.0002X'; // 11
stUserMotorData.stMotorData_ACIM.uiNominalSpeed := 1380;
stUserMotorData.stMotorData_ACIM.uiPolePair := 2;
stUserMotorData.stMotorData_ACIM.rMotorCosPhi := 0.70;
stUserMotorData.stMotorData_ACIM.uiNominalFrequency := 50;
stUserMotorData.stMotorData_ACIM.rNominalPower := 0.37;
stUserMotorData.stMotorData_ACIM.uiNominalVoltage := 400;
stUserMotorData.stMotorData_ACIM.rNominalCurrent := 1.14; // A
stUserMotorData.stMotorData_ACIM.rPeakCurrent :=
   stUserMotorData.stMotorData_ACIM.rNominalCurrent * 3.5;
stUserMotorData.stMotorData_ACIM.uiEncoderType :=
   MTP.ET_EncoderType.None; stUserMotorData.stMotorData_ACIM.uiTempSen-
sorType := 1; stUserMotorData.stMotorData_ACIM.uiMaxSpeed := 1920;
stUserMotorData.stMotorData_ACIM.uiBrake := 0;
stUserMotorData.stMotorData_ACIM.uiRotatingFieldDirection := 1;
stUserMotorData.stMotorData_ACIM.uiTempSensorResistanceOvertemp :=
   4000;
```

Section 5.4
ST_UserMotorDataPMSM

ST_UserMotorDataPMSM - General Information

Overview

Type:	Data structure
Available as of:	V1.0.0.0
Inherits from:	-

Description

This structure contains specific motor data for PMSM (Permanent Magnetic Synchronous Motors).

Structure Elements

Variable	Data type	Description
uiEncoderType	UINT	Encoder type (see ET_EncoderType *(see page 29)*) • 1: SinCos Hiperface encoder (rotary) • 2: SinCos encoder (rotary) • 257: SinCos Hiperface encoder (linear) • 258: SinCos encoder (linear) Required parameter
uiNominalSpeed	UINT	Rated velocity in [rpm] or [mm/s]. Required parameter
rNominalVoltage	REAL	Rated voltage in [Vrms]. rNominalVoltage = PhaseVoltage * root3 Required parameter
rPeakCurrent	REAL	Peak current of the motor in [Arms]. Required parameter
rNominalCurrent	REAL	Rated current of the motor at rated speed in [Arms]. If the technical data sheet of the motor does not distinguish the rated current at standstill and the rated current at rated speed, then the same value must be defined for rNominalCurrent and rContStallCurrent. Required parameter

Variable	Data type	Description
rContStallCurrent	REAL	Rated current of the motor at standstill in [Arms.] If the technical data sheet of the motor does not distinguish the rated current at standstill and the rated current at rated speed, then the same value must be defined for rNominalCurrent and rContStallCurrent. Required parameter
uiPolePair	UINT	Number of pairs of poles. Required parameter for rotary motors.
rConstStallTorque	REAL	Rated torque of the motor in [Nm] or nominal force of the motor in [N]. Required parameter
rPeakTorque	REAL	Peak torque of the motor in [Nm] or peak force of the motor in [N]. Required parameter
rPhaseResistance	REAL	Winding resistance of the motor (phase to phase) in [Ω]. Required parameter
rQuadraturePhaseInductance	REAL	Winding inductance (q component) of the motor (phase to phase) in [µH]. If the technical data sheet of the motor does not distinguish the d and q component, then the same value must be defined for rDirectPhaseInductance and rQuadraturePhaseInductance. Required parameter
uiRotatingFieldDirection	UINT	Direction of the rotary field (default value: 0) ● 0: counterclockwise ● 1: clockwise If the motor phases, ● U -> U ● V -> V ● W ->W are connected to the drive; then this parameter must be set to the value 1 (clockwise). Not required
uiMaxSpeed	UINT	Maximum speed of rotation of the motor in [rpm] or [mm/s]. Required parameter
udiMotorIntertia	UDINT	Mass inertia of the motor in [gcm²] for rotary motors or mass of the motor [g] for linear motors. Required parameter
uiBrake	UINT	Indicates whether a brake is integrated in the motor: ● 0: no brake integrated ● 1: brake integrated Required parameter

Variable	Data type	Description
`uiTempSensorType`	UINT	Type of the thermal sensor: ● 0: no sensor ● 1: nonlinear sensor (for example, PTC) ● 2: linear sensor (for example, KTY) Required parameter
`uiMaxMotorTemperature`	UINT	Maximum temperature of the motor in [°C] (default value: 130 °C) Not required
`uiTempSensorResistance Overtemp`	UINT	Minimum resistance of the temperature sensor if the temperature is too high in [Ω] (default value: 4 kΩ). The parameter can only be used for PTC sensors. `FC_MotorDataFileCreate` calculates the sensor characteristics depending on this parameter. The value is used only if the type of the temperature sensor is 1. Not required
`auiTempSensorCharacteristics`	ARRAY [0..19] OF UINT	Characteristic curve of the temperature sensor. This parameter is used only in the case of a linear sensor. In this case, you must specify all values: ● `auiTempSensorChracteristics[0]`: resistance in [Ω] for -30 °C ● `auiTempSensorChracteristics[1]`: resistance in [Ω] for -20 °C ● `auiTempSensorChracteristics[2]`: resistance in [Ω] for -10 °C ● ... ● `auiTempSensorChracteristics[19]`: resistance in [Ω] for 160 °C Not required
`rInsulationSystemVoltage`	REAL	Insulation voltage of the motor in [V]. The insulation voltage of the motor is not related to the DC bus voltage but to the maximum voltage which can occur in a motor phase. A voltage in the DC bus can be converted to a voltage in a motor phase as follows: `U_Motorphase := (U_Dcbus * 1.15) / (1.4142 * 2)` Example: The maximum permitted DC bus voltage for a motor is 500 V. The maximum permitted voltage in a motor phase is then: = (500 V * 1.15) / (1.4142 * 2) = 203. Not required
`uiEncoderMaxSpeed`	UINT	Maximum speed of rotation of the encoder [rpm]. Not required
`uiEncoderMaxTemp`	UINT	Maximum temperature of the encoder in [°C]. Not required

Variable	Data type	Description
uiEncoderTempSensor	UINT	Indicates whether the encoder has an integrated temperature sensor. ● 0: no temperature sensor ● 1: temperature sensor Not required
uiEncoderNumberOfTurns	UINT	Number of the displayable revolutions of the encoder. Required parameter for SinCos encoder (without Hiperface interface).
uiEncoderLinesPerrevolution	UINT	Number of encoder periods per revolution. Required parameter for SinCos encoder (without Hiperface interface).
rDirectPhaseInductance	REAL	Winding inductance (d component) of the motor (phase to phase) in [µH]. If the technical data sheet of the motor does not distinguish the d and q component, then the same value must be defined for rDirectPhaseInductance and rQuadraturePhaseInductance. Required parameter
rEMK_Constant	REAL	EMF (electromotive force) constant (phase to phase) in [Vrms / 1000 rpm] or [Vrms / m/s]. **NOTE:** In order to convert the EMF constant for one phase (string value) into a phase to phase value, the following formula must be used. `EMF_Constant_Phase_to_Phase := EMF_Constant_Phase * SQRT(3);` Required parameter
uiModelC1	UINT	Parameter C1 for the thermal model. Not required
uiModelA12	UINT	Parameter A12 for the thermal model. Not required
uiModelDeltaA	UINT	Parameter DeltaA for the thermal model. Not required
uiModelDeltaB	UINT	Parameter DeltaB for the thermal model. Not required
wBrakeType	WORD	Specifies the type of the brake if present (default value: 0). 0: default holding brake 1: inverted holding brake Not required
uiBrakeDisconnectionTime	UINT	Period until the brake is disengaged in [ms]. If uiBrakeDisconnectionTime is equal to 0, the drive uses the default value 100 ms. Not required

Variable	Data type	Description
`uiBrakeCouplingTime`	UINT	Period until the brake is engaged in [ms]. If `uiBrakeCouplingTime` is equal to 0, the drive uses the default value 100 ms. Not required
`rBrakeMinVoltage`	REAL	Minimum control voltage that is required to use the brake in [V]. If `rBrakeMinVoltage` is equal 0, then the brake voltage of the drive is not monitored. Not required
`rBrakeMaxVoltage`	REAL	Maximum control voltage that is permitted to use the brake in [V]. If `rBrakeMaxVoltage` is equal 0, then the brake voltage of the drive is not monitored. Not required
`uiBrakeNomCurrent`	UINT	Rated current of the brake in [mA]. Not required
`uiThermalConstant`	UINT	Thermal constant for monitoring the motor overheating in [ms]. The parameter defines for how long the motor can be operated at peak current without being overloaded. The default value is 1000 ms. Not required
`rPeriodLength`	REAL	Period length of the encoder in [µm]. Required parameter for linear motors.
`rPolePairPitch`	REAL	Pole pair distance in [µm]. Required parameter for linear motors.

Example

In the following example, a linear synchronous motor is configured. The motor has the following data:

- Rated velocity = 3200 mm/s
- Rated voltage = 600 V
- Rated current = 9.4 A
- Peak current = 20.7 A
- Rated force = 5 N
- Peak force = 200 N
- Winding resistance = 4.34 Ω
- Winding inductance = 32 mH
- EMF constant = 920 V / m/s
- Period length = 5 mm
- Pole pair distance = 2.4 mm
- Motor mass = 6.3 kg
- The motor has a linear SinCos Hiperface encoder.

- The motor has a nonlinear temperature sensor. If the motor temperature is too high, then the temperature sensor has a resistance > 4 kΩ.
- The motor has no brake.

The motor data is integrated into the data structure as follows:

```
stUserMotorData.eMotorType := Linear_PMSM;
stUserMotorData.sMotorname := 'Tec-TL30_N';
stUserMotorData.sMotorArticleNumber := 'ETYP00056';
stUserMotorData.sMotorSerialNumber := '201138.00';
stUserMotorData.stMotorDataPMSM.uiEncoderType :=
   MTP.ET_EncoderType.SincosHiperfaceLinear;
stUserMotorData.stMotorDataPMSM.uiPeriodLength := 5000; // µm
stUserMotorData.stMotorDataPMSM.uiNominalSpeed := 3200; // mm/s
stUserMotorData.stMotorDataPMSM.uiNominalVoltage := 600; // V
stUserMotorData.stMotorDataPMSM.rNominalCurrent := 9.4; // A
stUserMotorData.stMotorDataPMSM.rPeakCurrent := 20.7; // A
stUserMotorData.stMotorDataPMSM.rContStallCurrent := 9.4; // A
stUserMotorData.stMotorDataPMSM.rConstStallTorque := 5; // N
stUserMotorData.stMotorDataPMSM.rPeakTorque := 200; // N
stUserMotorData.stMotorDataPMSM.rPhaseResistance := 4.34; // Ohm
stUserMotorData.stMotorDataPMSM.rQuadraturePhaseInductance :=
   32000; // uH
stUserMotorData.stMotorDataPMSM.rDirectPhaseInductance := 32000; // uH
stUserMotorData.stMotorDataPMSM.uiRotatingFieldDirection := 1;
stUserMotorData.stMotorDataPMSM.udiEMK_Constant := 920; // V / m/s
stUserMotorData.stMotorDataPMSM.uiPolePairPitch := 2400; // µm
stUserMotorData.stMotorDataPMSM.uiMaxMotorTemperature := 130; //°C
stUserMotorData.stMotorDataPMSM.uiTempSensorType := 1;
stUserMotorData.stMotorDataPMSM.uiMaxSpeed := 3200; // mm/s
stUserMotorData.stMotorDataPMSM.udiMotorIntertia := 6300; // g
stUserMotorData.stMotorDataPMSM.uiBrake := 0;
stUserMotorData.stMotorDataACIM.uiTempSensorResistanceOvertemp := 4000;
```

Chapter 6
Global Elements

Section 6.1
GCL (Global Constant List)

GCL (Global Constant List) - General Information

Overview

Type:	Global constants
Available as of:	V1.0.0.0

Description

GCL (Global Constant List) of the UserMotorTypePlate library.

Global Constants

Variable	Data type	Value	Description
Gc_sLibraryVersion	STRING [80]	Vx.x.x.x	Library version.
Gc_byMotorTypePlateVersion	BYTE	2	Motor type plate version.
Gc_byMotorTypePlateRevision	BYTE	29	Motor type plate revision.
Gc_wMotorTypePlateSubrevision	WORD	1	Motor type plate subrevision.
Gc_uiEncoderMaxSpeedMinValue	UDINT	0	Minimum value of input parameter EncoderMinSpeed.
Gc_uiEncoderMaxSpeedMaxValue	UDINT	65534	Maximum value of input parameter EncoderMaxSpeed.
Gc_uiEncoderMaxTempMinValue	UDINT	0	Minimum value of input parameter EncoderMinTemp.
Gc_uiEncoderMaxTempMaxValue	UDINT	65534	Maximum value of input parameter EncoderMinTemp.
Gc_uiEncoderTempSensorMinValue	UDINT	0	Minimum value of input parameter EncoderTempSensor.
Gc_uiEncoderTempSensorMaxValue	UDINT	1	Maximum value of input parameter EncoderTempSensor.
Gc_uiEncoderNumberOfTurnsMinValue	UDINT	0	Minimum value of input parameter EncoderNumberOfTurns.
Gc_uiEncoderNumberOfTurnsMaxValue	UDINT	65534	Maximum value of input parameter EncoderNumberOfTurns.
Gc_uiEncoderLinesPerRevolutionMinValue	UDINT	0	Minimum value of input parameter EncoderLinesPerRevolution.
Gc_uiEncoderLinesPerRevolutionMaxValue	UDINT	65534	Maximum value of input parameter EncoderLinesPerRevolution.

Glossary

B

BSH

A Lexium servo motor from Schneider Electric.

byte

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

C

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

E

encoder

A device for length or angular measurement (linear or rotary encoders).

F

firmware

Represents the BIOS, data parameters, and programming instructions that constitute the operating system on a controller. The firmware is stored in non-volatile memory within the controller.

function block

A programming unit that has 1 or more inputs and returns 1 or more outputs. FBs are called through an instance (function block copy with dedicated name and variables) and each instance has a persistent state (outputs and internal variables) from 1 call to the other.

Examples: timers, counters

P

POU

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

S

Sercos

(*serial real-time communications system*) A digital control bus that interconnects, motion controls, drives, I/Os, sensors, and actuators for numerically controlled machines and systems. It is a standardized and open controller-to-intelligent digital device interface, designed for high-speed serial communication of standardized closed-loop real-time data.

Index

F

G

I

L

M

N

O

P

Q

R

S

U

W