



# Manual



## **MOVIKIT® Power and Energy Solutions** DataLogging



## Table of contents

<b>1</b>	<b>General information.....</b>	<b>5</b>
1.1	About this documentation .....	5
1.2	Content of the documentation.....	5
1.3	Structure of the safety notes .....	5
1.3.1	Meaning of signal words .....	5
1.3.2	Structure of section-related safety notes.....	5
1.3.3	Structure of embedded safety notes .....	6
1.4	Decimal separator in numerical values .....	6
1.5	Rights to claim under limited warranty .....	6
1.6	Product names and trademarks.....	6
1.6.1	Trademark of Beckhoff Automation GmbH .....	6
1.7	Copyright notice .....	6
1.8	Other applicable documentation .....	7
1.9	Short designation .....	7
<b>2</b>	<b>Safety notes .....</b>	<b>8</b>
2.1	Preliminary information .....	8
2.2	Target group .....	8
2.3	Network security and access protection .....	8
2.4	Designated use .....	8
<b>3</b>	<b>System description .....</b>	<b>9</b>
3.1	Module description.....	9
3.2	Functions .....	9
3.3	Components.....	10
3.3.1	IEC libraries.....	11
<b>4</b>	<b>Project planning information.....</b>	<b>12</b>
4.1	Requirement .....	12
4.2	Hardware .....	12
4.3	Software.....	12
4.4	Licensing.....	12
<b>5</b>	<b>Startup .....</b>	<b>13</b>
5.1	Requirements.....	13
5.2	Startup procedure .....	13
5.3	Configuring a project.....	14
5.3.1	Example project .....	14
5.4	Generating an IEC project .....	15
5.4.1	IEC project structure .....	16
<b>6</b>	<b>IEC programming.....</b>	<b>17</b>
6.1	Opening the IEC project.....	17
6.2	Basic functions.....	17
6.2.1	Diagnostics.....	17
6.2.2	Access management.....	18
6.3	IEC libraries .....	19
6.3.1	SEW PES DataLogger .....	19

6.3.2 SEW PES EnergyMeter ..... 24

6.3.3 SEW PES RealTimeScope ..... 32

**7 Fault management..... 40**

7.1 Fault codes ..... 40

7.1.1 RealTimeScope..... 40

7.1.2 Data logger..... 40

**Index ..... 41**



# 1 General information

## 1.1 About this documentation

This documentation is an integral part of the product. The documentation is intended for all employees who perform work on the product.

Make sure this documentation is accessible and legible. Ensure that persons responsible for the systems and their operation as well as persons who work with the product independently have read through the documentation carefully and understood it. If you are unclear about any of the information in this documentation, or if you require further information, contact SEW-EURODRIVE.

## 1.2 Content of the documentation

The descriptions in this documentation apply to the software and firmware versions applicable at the time of publication. These descriptions might differ if you install later software or firmware versions. In this case, contact SEW-EURODRIVE.

## 1.3 Structure of the safety notes

### 1.3.1 Meaning of signal words

The following table shows the grading and meaning of the signal words for safety notes.

Signal word	Meaning	Consequences if disregarded
<b>▲ DANGER</b>	Imminent hazard	Severe or fatal injuries
<b>▲ WARNING</b>	Possible dangerous situation	Severe or fatal injuries
<b>▲ CAUTION</b>	Possible dangerous situation	Minor injuries
<b>NOTICE</b>	Possible damage to property	Damage to the product or its environment
<b>INFORMATION</b>	Useful information or tip: Simplifies handling of the product.	

### 1.3.2 Structure of section-related safety notes

Section-related safety notes do not apply to a specific action but to several actions pertaining to one subject. The hazard symbols used either indicate a general hazard or a specific hazard.

This is the formal structure of a safety note for a specific section:



#### **SIGNAL WORD**


Type and source of hazard.

Possible consequence(s) if disregarded.

- Measure(s) to prevent the hazard.

## Meaning of the hazard symbols

The hazard symbols in the safety notes have the following meaning:

Hazard symbol	Meaning
	General hazard

### 1.3.3 Structure of embedded safety notes

Embedded safety notes are directly integrated into the instructions just before the description of the dangerous action.

This is the formal structure of an embedded safety note:

**⚠ SIGNAL WORD!** Type and source of hazard. Possible consequence(s) if disregarded. Measure(s) to prevent the hazard.

## 1.4 Decimal separator in numerical values

In this document, a period is used to indicate the decimal separator.

Example: 30.5 kg

## 1.5 Rights to claim under limited warranty

Read the information in this documentation. This is essential for fault-free operation and fulfillment of any rights to claim under limited warranty. Read the documentation before you start working with the product.

## 1.6 Product names and trademarks

The brands and product names in this documentation are trademarks or registered trademarks of their respective titleholders.

### 1.6.1 Trademark of Beckhoff Automation GmbH

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.



## 1.7 Copyright notice

© 2020 SEW-EURODRIVE. All rights reserved. Unauthorized reproduction, modification, distribution or any other use of the whole or any part of this documentation is strictly prohibited.

## 1.8 Other applicable documentation

Observe the corresponding documentation for all further components.

Always use the latest edition of the documentation and the software.

The SEW-EURODRIVE website ([www.sew-eurodrive.com](http://www.sew-eurodrive.com)) provides a wide selection of documents for download in various languages. If required, you can also order printed and bound copies of the documentation from SEW-EURODRIVE.

## 1.9 Short designation

The following short designations are used in this documentation:

Type designation	Short designation
MOVIKIT® Power and Energy Solutions DataLogging	MOVIKIT® DataLogging
MOVIKIT® Power and Energy Solutions DataLogging	Software module
MDP92A power supply module with controlled DC link voltage	MDP92A
DC/DC converter module MDE90A	MDE90A

## 2 Safety notes

### 2.1 Preliminary information

The following general safety notes serve the purpose of preventing injury to persons and damage to property. They primarily apply to the use of products described in this documentation. If you use additional components, also observe the relevant warning and safety notes.

### 2.2 Target group

**Software specialist** Any work with the software may only be performed by a specialist with suitable training. A specialist in this context is someone who has the following qualifications:

- Appropriate training
- Knowledge of this documentation and other applicable documentation
- SEW-EURODRIVE recommends additional training for products that are operated using this software.

### 2.3 Network security and access protection

A bus system makes it possible to adapt electronic drive technology components to the particulars of the machinery within wide limits. There is a risk that a change of parameters that cannot be detected externally may result in unexpected but not uncontrolled system behavior and may have a negative impact on operational safety, system availability, or data security.

Ensure that unauthorized access is prevented, especially with respect to Ethernet-based networked systems and engineering interfaces.

Use IT-specific safety standards to increase access protection to the ports. For a port overview, refer to the respective technical data of the device in use.

### 2.4 Designated use

The MOVIKIT® Power and Energy Solutions DataLogging is used in conjunction with the hardware components of the Power and Energy Solutions product range to provide intelligent power and energy management.

Use the device-independent MOVISUITE® engineering software to start up and configure the devices and to download the complete configuration to a MOVI-C® CONTROLLER.

Observe the documentation for the components used.

Unintended or improper use of the product may result in severe injury to persons and damage to property.

## 3 System description

### 3.1 Module description

Introduction	The Power and Energy Solutions product range expands SEW-EURODRIVE's current line of MOVIDRIVE® modular inverters to include intelligent power and energy management components. In addition to hardware components, the product range includes the MOVIKIT® modular system with software modules for optimum integration at the control software level.
MOVIKIT® DataLogging	MOVIKIT® DataLogging offers functions for logging variables of the application and also offers an energy meter and a real-time oscilloscope.

### 3.2 Functions

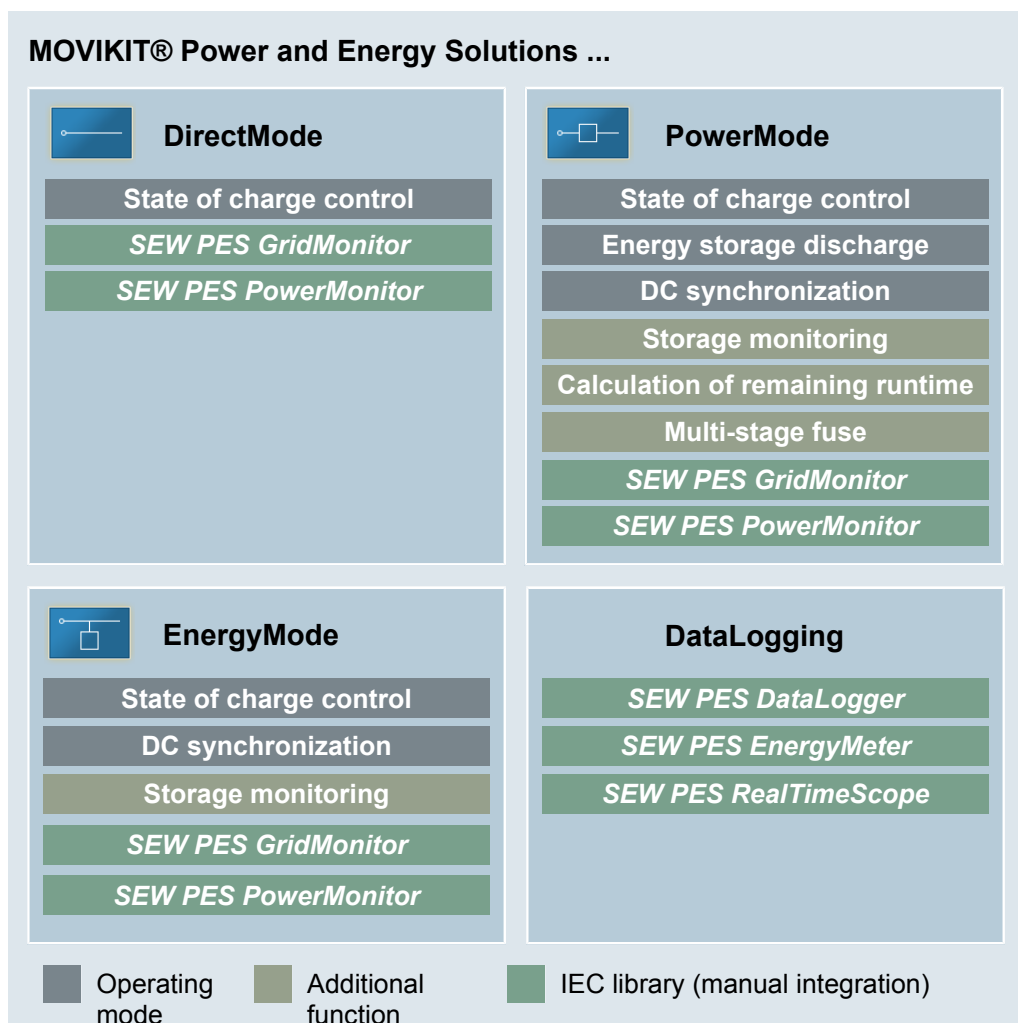
Overview of functions:

- **Recording of power and energy data**
  - Flexible energy counters directly on the MOVI-C® CONTROLLER
  - Real-time scope for data recording and transfer to a Windows system
  - DataLogger for recording long-term data and transfer to a Windows system



### 3.3 Components

The following figure provides an overview of the software modules of the MOVIKIT® Power and Energy Solutions modular system:



9007230506411147

### 3.3.1 IEC libraries

MOVIKIT® DataLogging also provides the following IEC libraries for manual integration:

- **SEW PES EnergyMeter**  
Provides various types of energy meters.
- **SEW PES RealTimeScope**  
Real time scope (similar to an oscilloscope) for recording up to 8 channels and providing a CSV file for transfer to a Windows system
- **SEW PES DataLogger**  
Block for recording any variables of the application of the REAL data type and providing them as CSV file for transfer to a Windows system

For further information, refer to chapter "IEC libraries" (→ 19).

## 4 Project planning information

### 4.1 Requirement

Correct project planning and proper installation of the devices are required for successful startup and operation.

For detailed project planning information, refer to the documentation of the respective devices.

### 4.2 Hardware

The following hardware is required:

- MOVI-C® CONTROLLER (recommended for UHX45A performance class and higher)

### 4.3 Software

The following software is required:

- MOVISUITE® engineering software  
(includes MOVIRUN® flexible)

For more detailed information on the hardware requirements of the individual software components, see the documentation for the respective software.

### 4.4 Licensing

The following licenses are available and are required:

- MOVIRUN® flexible

License for the software platform MOVIRUN® flexible

For further information on licensing, refer to the document "MOVI-C® Software Components". You can download the document from the SEW-EURODRIVE website ([www.sew-eurodrive.com](http://www.sew-eurodrive.com)).

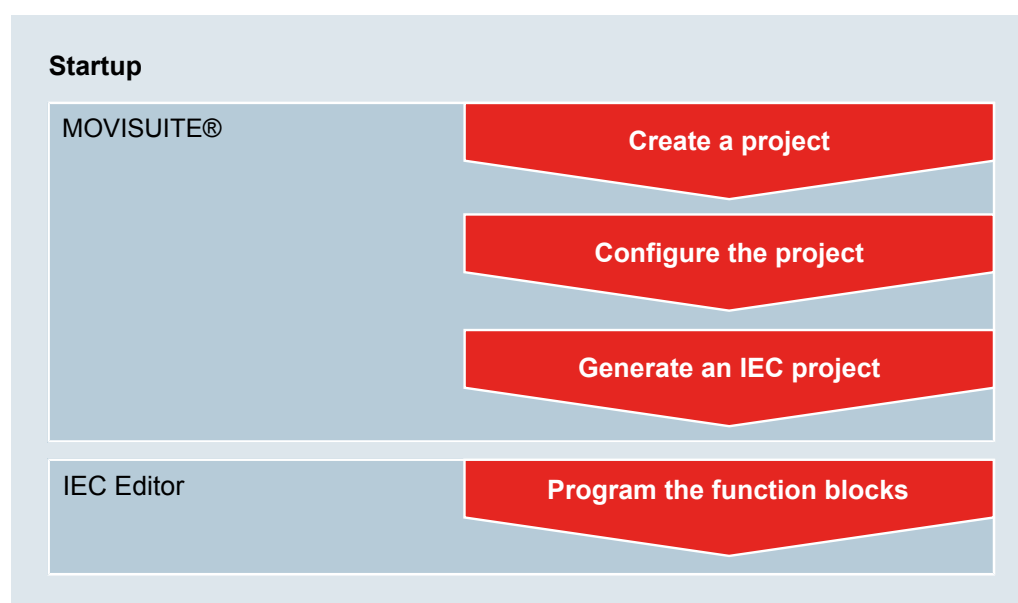
## 5 Startup

### 5.1 Requirements

- Check the installation of the inverters and, if installed, also check the encoder connection.
- Observe the installation notes in the documentation of the respective device and software components.
- The devices to be started up are displayed in MOVISUITE®.

### 5.2 Startup procedure

The schematic diagram below shows the startup procedure:



18014426855028235

The startup steps specific to these software modules are explained in detail in the following chapters of this manual. For startup, also observe the documentation of all the other components in use.

### 5.3 Configuring a project

#### INFORMATION

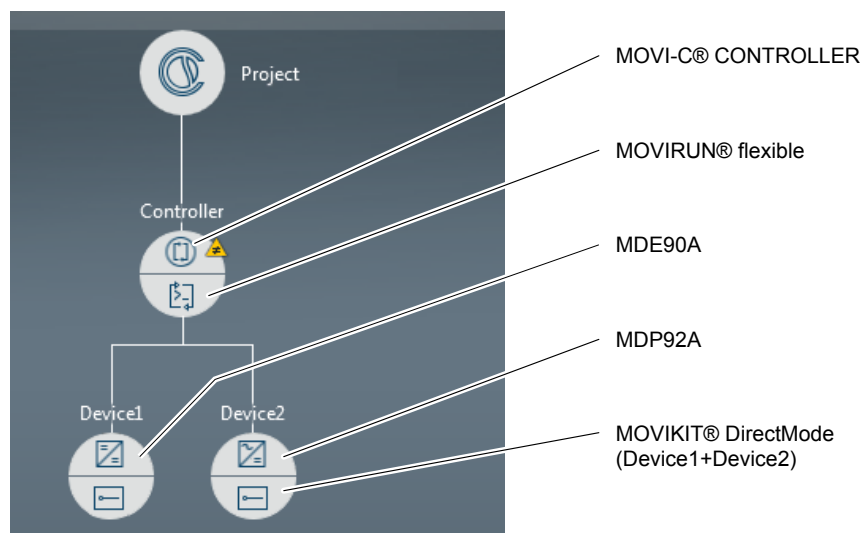


For detailed information on how to use the MOVISUITE® engineering software, refer to the corresponding documentation.

- ✓ A MOVISUITE® project has been created and is open.
- 1. Add required device nodes, software nodes (MOVI-C® SoftwareNode) and software modules to the project.
  - ⇒ See "Example project".
- 2. Configure the added devices or software modules. If available, observe the specific notes in the following chapters that apply to MOVIKIT® Power and Energy Solutions DataLogging. For detailed information on the configuration of devices or other software modules, refer to the respective documentation.

#### 5.3.1 Example project

The following figure shows an example project:



31237315339



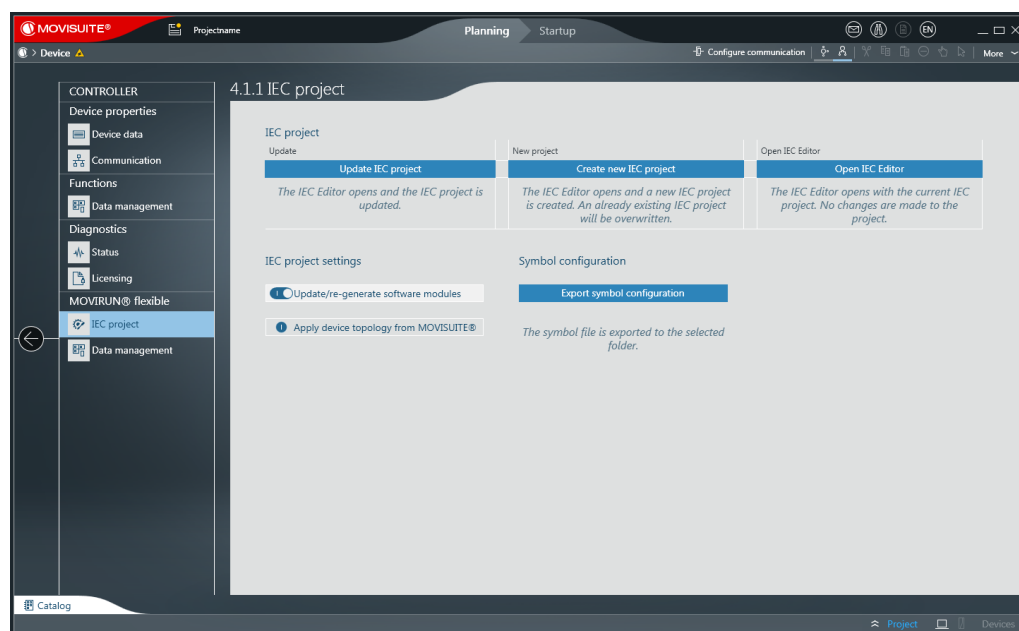
## 5.4 Generating an IEC project

Carry out the following steps to create an IEC project using automatic code generation and based on the configuration settings in MOVISUITE®.

✓ Configuration of the MOVISUITE® project has been completed.

1. In the function view of MOVISUITE®, click on the software module section of the MOVI-C® CONTROLLER.

⇒ The "IEC project" menu opens.



27021618448637067

### INFORMATION



If you have carried out the configuration in MOVISUITE® using the "Startup" mode and the message "Device cannot be reached" appears, proceed as follows:

- If the MOVI-C® CONTROLLER is not available via the network, switch over to "Planning" mode.
- If the MOVI-C® CONTROLLER is available via the network, carry out a network scan and connect the MOVI-C® CONTROLLER in the network view with the MOVI-C® CONTROLLER in the function view.

2. Click [Create new IEC project].

⇒ The IEC Editor opens and a new IEC project is created.

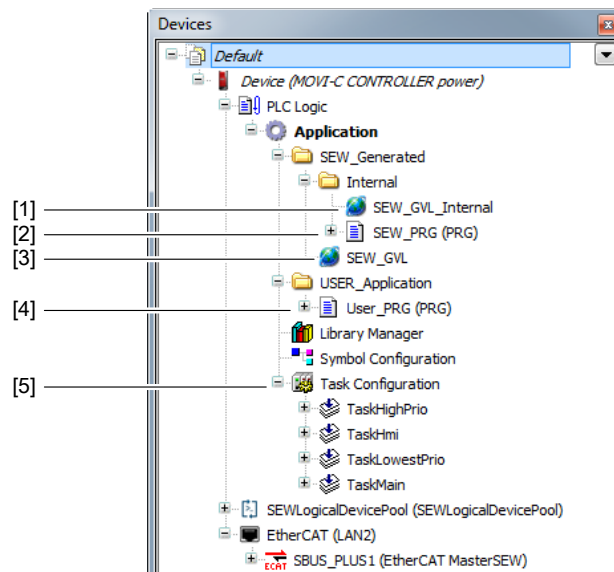
### INFORMATION



If changes are made to the project structure, to inverter data sets, or to a software module configuration after the IEC project is generated for the first time, a notification symbol is displayed on the MOVI-C® CONTROLLER node. Click on the message icon for more information about the change, and to update the IEC project.

### 5.4.1 IEC project structure

The IEC project has the following basic structure:



18014423003085323

No.	Name	Description
[1]	SEW_GVL_Internal	<p>The SEW_GVL_Internal global list of variables contains the instances that correspond to the software module used. These variables may not be written to from the user program.</p> <p>In addition, the structure contains an instance as a communication buffer for controlling or monitoring the software module by means of a monitor.</p>
[2]	SEW_PRG	<p>Program that contains all the important instance calls. Automatic code generation recreates this program in accordance with the configuration made in MOVISUITE® each time the IEC project is created, thereby overwriting the previous version. Therefore, you should not make any changes to this program.</p>
[3]	SEW_GVL	<p>The SEW_GVL global list of variables is the interface for accessing the software module features.</p>
[4]	User_PRG	<p>The user program is created once, initially, by automatic code generation. Since the program is not overwritten with each subsequent creation, this is the appropriate place for integrating user programs.</p> <p>The program is divided into five actions. These actions differ in the time at which they are called during the program sequence.</p>
[5]	Task configuration	<p>The list of tasks created in the project. Automatic code generation initially adds tasks that differ in how they are prioritized.</p> <p>The user can add additional programs to existing tasks or create new tasks.</p> <p>It is the responsibility of the user to design the capacity utilization of the tasks to enable the tasks to be processed within the required cycle time. Moving beyond the cyclical tasks, in particular, prevents setpoints for the interpolating axes from being generated in time, which means that these axes cannot be operated properly.</p>

## 6 IEC programming

### 6.1 Opening the IEC project

- If an IEC project has already been generated, select the [IEC Editor] entry under "Tools" from the context menu of the MOVI-C® CONTROLLER in MOVISUITE®.
- If no IEC project has been generated, follow the steps described in the "Generating an IEC project" (→ 15) chapter.

### 6.2 Basic functions

#### 6.2.1 Diagnostics

Variables for reporting and writing errors and warnings.

Variable name	Description
xError	Data type – BOOL
	<ul style="list-style-type: none"><li>• TRUE – Error present</li><li>• FALSE – No error present</li></ul>
xWarning	Data type – BOOL
	<ul style="list-style-type: none"><li>• TRUE – Warning present</li><li>• FALSE – No warning present</li></ul>
udiMessageID	Data type: UDINT
	Message ID number
sAdditionalText	Data type: STRING
	Additional message text
xReset	Data type – BOOL
	<ul style="list-style-type: none"><li>• TRUE – Reset messages</li><li>• FALSE – Do not reset messages</li></ul>

### 6.2.2 Access management

Variables for managing access permissions.

Variable name	Description
xGetAccessControl	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Request access</li> <li>• FALSE – Return access</li> </ul>
xControlActive	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Access granted</li> <li>• FALSE – Access denied</li> </ul>

#### Access via user interface (UserInterface):

An instance requests access by setting *xGetAccessControl* to "TRUE". If *xControlActive* returns a "TRUE" value, access has been granted and is now permitted.

#### Access via other function blocks in the user program:

If another function block in control mode wants to access the device interface at the same time as the user interface (UserInterface), the device interface decides which one receives write permission based on the priority of the function block. The user interface (UserInterface) has the highest priority. This means if the user interface (UserInterface) has control access, then *xControlActive* returns "FALSE" to all other function blocks.

## 6.3 IEC libraries

### 6.3.1 SEW PES DataLogger



#### INFORMATION

The function block is not integrated via automatic code generation but must be implemented manually in the project.

The *SEW PES DataLogging* library provides a function for permanent logging of slowly changing data.

#### Operating principle

The *DataLogger* function block records the data on the MOVI-C® CONTROLLER. The data is saved in a CSV file on a Windows system (e.g. Windows part of the MOVI-C® CONTROLLER or external PCs). For storage, the function block provides a buffer variable that is used to exchange one line of the CSV file at a time. Also included is another variable for the handshake to control the transfer. Data is exchanged via a standardized communication protocol such as CODESYS ARTI.

Various variables are provided in the *logging* structure for transferring the CSV file from the MOVI-C® CONTROLLER to the connected Windows system. See chapter "Logging" (→ 22).

After each recorded sample, a line of the CSV file is provided for transfer. In each cycle, the *DataLogger* function block first checks the status of the variable *IN-OUT.xNewData*. If this variable has the value "FALSE", the receiver side is ready. In this case, one line of the CSV file is written to the variable *OUT.abvLineBuffer*. The variable *INOUT.xNewData* is then set to "TRUE" to signal to the receiver side that a new row is ready for transfer. As soon as the receiver side has fetched the line, it must set the variable *INOUT.xNewData* to "FALSE" to signal to the function block that the next line can be transferred. This procedure is repeated until the *IN.xActivate* variable is set to "FALSE".



## Integration

Do the following to integrate the functionality:

1. Create an instance of the *DataLogger* function block (e.g. in the *User\_PRG* program or in a global variable list)

```
⇒ fbDataLogger : SEW_PES_DataLogger.DataLogger;
```

2. Initialize the *DataLogger* function block in the *Init* action of the *User\_PRG* program, assign the recording channels and perform the basic settings for the measurement. The following code block example shows the initialization of the *DataLogger* function block in the *Init* action of the *User\_PRG* program as well as the assignment of the recording channels and the basic settings for the measurement.

```
⇒ //Initialization of function block
   IF NOT xInitDoneDataLogger THEN
     xInitDoneDataLogger:=fbDataLogger.Init(
       diCycleTime:=ANY_TO_DINT(gc_uiTaskCycleTime)
     );
   END_IF

   //Default settings

   //Assignment of channels
   fbDataLogger.MapChannel(0,ADR(rDlcTemp),'T_DLC [°C]');
   fbDataLogger.MapChannel(1,ADR(rMotCycCnt),'Motion Cycle');

   //Activation of channels
   fbDataLogger.Logging.IN.axEnable[0]:=TRUE;
   fbDataLogger.Logging.IN.axEnable[1]:=TRUE;
   fbDataLogger.Logging.IN.axEnable[2]:=FALSE;
   fbDataLogger.Logging.IN.axEnable[3]:=FALSE;
   fbDataLogger.Logging.IN.axEnable[4]:=FALSE;
   fbDataLogger.Logging.IN.axEnable[5]:=FALSE;
   fbDataLogger.Logging.IN.axEnable[6]:=FALSE;
   fbDataLogger.Logging.IN.axEnable[7]:=FALSE;

   //Time base settings
   fbDataLogger.Logging.IN.tIntervalTime:=T#5s;

   //Activation of DataLogger
   fbDataLogger.Logging.IN.xActivate:=TRUE;
```

3. In the *Main* action of the *User\_PRG* program, call the *CallMain* method of the instance of the *DataLogger* function block.

```
⇒ fbDataLogger.CallMain();
```

4. In the *HighPrio* action in the *User\_PRG* program, call the *CallHighPrio* method of the *DataLogger* function block instance.

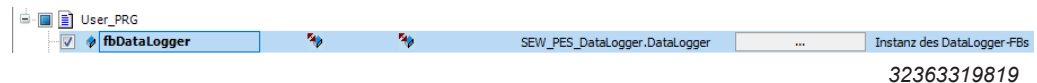
```
⇒ fbDataLogger.CallHighPrio();
```

- ⇒ When you log on to the MOVI-C® CONTROLLER, you can now display the variables by double-clicking the *User\_PRG* program or the selected global variable list in the device tree under the instance of the *DataLogger* function block.

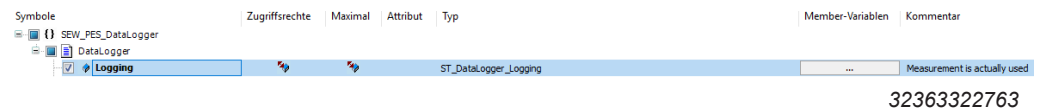
## Symbol configuration

Non-volatile data recorded with the function block must be stored externally. A Windows system is required for this purpose, which is either located on the Windows part of the MOVI-C® CONTROLLER (if available) or on a PC connected to the MOVI-C® CONTROLLER via the engineering interface. To transfer the recorded CSV file to the Windows system via CODESYS ARTI, the required variables must be made known to the Windows system. Proceed as follows:

1. Create a symbol configuration in the application program.
2. Specify the instance of the function block in the symbol configuration selection window.



3. Click on the button [...] in the "Member variables" column.  
⇒ A dialog box for selecting the function block member variables to be exported is opened.
4. Select the function block and confirm your selection with [OK].



5. Click on [Generate Code] under [Create] on the menu bar.  
⇒ The symbol configuration is exported as an XML file.
6. Import the generated XML file into the application on the Windows system that retrieves and saves the lines of the CSV file from the MOVI-C® CONTROLLER.

## Methods

### Init

Method for initializing the function block. The method must be called in the *Init* action of the *User\_PRG* program.

Variable name	Description
diCycleTime	Data type - DINT Cycle time of the <i>HighPrio</i> task of the program in [ms]

### CallMain

Method for running the program components of the block that are to be processed cyclically in the free-running task of the application program. The method must be called in the *Main* action of the *User\_PRG* program.

### CallHighPrio

Method for running the program components of the block that are to be processed cyclically in the real-time task of the application program. The method must be called in the *HighPrio* action of the *User\_PRG* program.

*MapChannel*

Method for assigning the channels of the block to the variables to be logged.

Variable name	Description
diChannel	Data type - DINT
	Number of the channel to which a variable is to be assigned.
prVariable	Data type – POINTER TO REAL
	Pointer to the variable to which the channel is to be assigned.
sCaption	Data type: STRING(127)
	Identifier of the signal for the header of the .CSV file to be saved

*SetTimeStampCaption*

Method for assigning an identifier of the time stamp column to the header of the CSV file to be saved.

Variable name	Description
sTimeStampCaption	Data type: STRING(127)
	Identifier of the time stamp column for the header of the CSV file to be saved

**Diagnostics**

The variables available in this structure are described in chapter "Diagnostics" (→ 17).

**Logging**

Structure for handling the *DataLogger* function block, which means to activate and deactivate the function and to output status information.

*IN*

Variable name	Description
axEnable	Data type – ARRAY [0..7] OF BOOL
	<p>Individually specify whether the data of channels 0 to 7 is to be recorded. The following applies to each channel:</p> <ul style="list-style-type: none"> <li>• TRUE – Record data of the channel</li> <li>• FALSE – Do not record data of the channel</li> </ul> <p>The prerequisite for recording the data of a channel is that it has been previously assigned correctly using the <i>MapChannel()</i> method.</p>

Variable name	Description
tIntervalTime	Data type – TIME Time interval between two measured values in a CSV file. The unit is specified by the user. Converted into [ms], the value must be an integer divisible by the cycle time of the <i>HighPrio</i> task and be at least 1 s in size.
xActivate	Data type – BOOL <ul style="list-style-type: none"> <li>• TRUE – Activate function</li> <li>• FALSE – Deactivate function</li> </ul>

## OUT

Variable name	Description
xActive	Data type – BOOL <ul style="list-style-type: none"> <li>• TRUE – Function active</li> <li>• FALSE – Function inactive</li> </ul>
abyLineBuffer	Data type - ARRAY [0..GVL.gc_diLineBufSize-1] OF BYTE Buffer that provides the current line of the CSV file to be transferred

## INOUT

Variable name	Description
xNewData	Data type – BOOL Handshake variable for transferring the CSV file. <ul style="list-style-type: none"> <li>• TRUE - The function block has provided a new line for transfer in <i>abyLineBuffer</i>.</li> <li>• FALSE - No transfer active. The function block can provide a new line in <i>abyLineBuffer</i> for transfer.</li> </ul>

## 6.3.2 SEW PES EnergyMeter

## INFORMATION



The function block is not integrated via automatic code generation but must be implemented manually in the project.

The *SEW PES EnergyMeter* IEC library provides energy meters with different levels of functionality.

## Operating principle

The energy meters calculate the energy from the input power consumed by the counting function (*AddPositivePwr*, *AddNegativePwr* or *AddPwr*) and from the call frequency. The calculated value is added to the actual value and the result value is output via a counter variable (*rEAct\_InWs*, *rEAct\_Cmpl\_InWs*, *rEAct\_Neg\_InWs* or *rEAct\_Pos\_InWs*).

## Counting variables

The following counting variables are available:

- *AddPositivePwr*

Counts only positive power values. Negative power values are ignored (return inhibited).

- *AddNegativePwr*

Counts only negative power values. Positive power values are ignored (advance inhibited).

- *AddPwr*

Counts negative and positive power values, taking into account the sign. This means that negative energy values can also result and a positive energy balance can be distinguished from a negative energy balance.

## Memory

The function blocks of the energy meters themselves have no remanent memory. In order to avoid losing information when transferring the initial value during initialization, the output can be connected to a remanent memory and its value can be used as the initial value.

## Implementation

All blocks of the energy meters must be cyclically supplied with power. Since the counting function records the power value, the appropriate counting function must run in a synchronous task or be called synchronously. Maximum accuracy is achieved if this synchronous task simultaneously operates the fieldbus or is connected to it. If a lower resolution is selected, the call cycle of the counting function should be a multiple of the task that provides the power value.



## Integration

Do the following to integrate the functionality:

1. Declare the energy meter in a suitable location.

⇒ `stEnergy:EnergyCounter;`

2. Initialize the function block using the *Init* function.

⇒ `xInitDone:=stEnergy.Init(  
pr_InWs_PointerSet:=ADR(rStartVal),  
rTimeBaseIn_sSet:=0.001,  
sFileName:='TestSav'  
);`

- ⇒ The function block is initialized with the time interval of the time-synchronous task, which later calls the counting function, with an initial value and with the name of the file where the counter values are stored.

## INFORMATION



If no remanent memory is used to determine the initial value, it should be initialized with the value 0. The *Init* function can be called repeatedly. The transferred changes become active immediately.

3. Transfer the current power value to the energy meter in a cyclically called method (such as *User\_PRG.HighPrio*):

⇒ `stEnergy.AddPwr(rActPower:=rActPower);`

4. Set the *xEnable* variable to "TRUE."

⇒ `stEnergy.xEnable:=TRUE;`

- ⇒ The energy meter is activated and the counting function starts logging values.

- ⇒ The values are reset by setting *xReset* to "TRUE."

## Single counters

"Single counters" are energy meters that have **one** counting variable. This means that **one** counting function is used if this energy meter is used.

### EnergyCounter

The *EnergyCounter* single counter is initialized with the call rate/sample rate and increments the energy value ( $E=P \cdot t$ ) each time the *Add\*Pwr* counting function is called.

Variables to start the energy meter and to output the actual energy value.

Variable name	Description
xEnable	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Start energy meter</li> <li>• FALSE – Stop energy meter</li> </ul>
rAct_InWs	Data type – REAL
	Current energy value in [W]

#### Note:

The function block also provides variables for "Access management" (→ 18) and "Diagnostics" (→ 17).

*EnergyIntervalCounter*

The *EnergyIntervalCounter* single counter offers the functionality of the *EnergyCounter* single counter as well as an interval function for defining a logging interval.

The *rEAct\_InWs* counter variable can be modified externally for the *EnergyIntervalCounter* single counter. In this way, a new initial value can be specified after initialization or during operation.

To use the interval functions, the logging interval must be set once using the *SetInterval()* function. The logging interval can be set or changed as often as desired.

The *EnergyIntervalCounter* single counter has the following operating modes, which are activated via the variables *xSingleShot* and *xStartStop* as described in the following list:

- Free-running: *xStartStop* = "FALSE", *xSingleShot* = "FALSE"

The *rEInterval* variable outputs the total of the last interval. The *rEAct\_InWs* variable outputs the counter reading of the current interval. A trigger signal is ignored.

- Single shot: *xStartStop* = "FALSE", *xSingleShot* = "TRUE"

The variables *rEAct\_InWs* and *rEInterval* stop and wait for a positive edge of the *xTrigger* variable. If the *xTrigger* variable has the value "TRUE," then *rEAct\_InWs* and *rEInterval* are set to 0 and *rEAct\_InWs* starts to count. If the interval expires, the *rEAct\_InWs* variable transfers its value to the *rEInterval* variable and is reset. As a result, the *rEInterval* variable contains the energy from the interval starting from the trigger time. The value is valid if the *xRunning* variable has the value "FALSE" and *xSingleShot* has the value "TRUE." An additional measurement is possible only if *xSingleShot* has a positive edge.

- Normal: *xStartStop* = "TRUE", *xSingleShot* = "FALSE"

The variables *rEAct\_InWs* and *rEInterval* stop and wait for a positive edge of the *xTrigger* variable. If the *xTrigger* variable has the value "TRUE", the *rEAct\_InWs* counter is set to 0. The *rEInterval* variable stops. If the interval expires, the *rEAct\_InWs* variable transfers its value to the *rEInterval* variable and is reset. As a result, the *rEInterval* variable always contains the energy of the last interval. Each interval begins with a positive edge of the *xTrigger* variable. The *xStartStop* variable does not have to be reset for the next interval. This operating mode is the same as the "normal" operating mode of the oscilloscope.

- Incorrect operation: *xStartStop* = "TRUE", *xSingleShot* = "TRUE"

A special case that should not occur. In this case, the function block also behaves like an oscilloscope and gives preference to the single shot (see above).

Variables for handling the energy meter, i.e. to activate and deactivate the function and to output status information.

Variable name	Description
xEnable	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Start energy meter</li> <li>• FALSE – Stop energy meter</li> </ul>
xReset	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Reset messages</li> <li>• FALSE – Do not reset messages</li> </ul>

Variable name	Description
xError	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Error present</li> <li>• FALSE – No error present</li> </ul>
xWarning	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Warning present</li> <li>• FALSE – No warning present</li> </ul>
udiMessageID	Data type: UDINT
	Message ID number
sAdditionalText	Data type: STRING
	Additional message text
xGetAccessControl	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Request access</li> <li>• FALSE – Return access</li> </ul>
xControlActive	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Access granted</li> <li>• FALSE – Access denied</li> </ul>
rEAct_InWs	Data type – REAL
	Current energy value in [W]
xResetCounter	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE - Reset counter reading.</li> <li>• FALSE - Do not reset counter reading.</li> </ul>
xRunning	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Recording in progress</li> <li>• FALSE – Recording not in progress</li> </ul>
xSingleShot	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Activate "single measurement" operating mode.</li> <li>• FALSE – Deactivate "single measurement" operating mode.</li> </ul>
xStartStop	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Activate "normal" operating mode. (Similar to the function of the storage unit oscilloscope)</li> <li>• FALSE – Deactivate "normal" operating mode.</li> </ul>
xTrigger	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Activate triggering at positive edge</li> <li>• FALSE – Do not activate triggering</li> </ul>
rEInterval	Data type – BOOL
	Total energy of the last measurement. The actual value is not available until the current measurement is completed

## Multi-counter

"Multi-counters" are energy meters that contain 3 single counters. One single counter counts positive, one negative, and one bidirectional. The energy balance, energy consumption and energy flow are always available separately from each other via the values of the 3 single counters. The 3 single counters can be initialized with energy values stored in the non-volatile memory.

"Multi-counter" energy meters can store the determined values in the persistent memory on the memory card. If the energy values are initially set to 0 and a file name is transferred during initialization, an attempt is made to start with the stored value. If the file does not yet exist, it is created and filled with the actual values (usually 0). If the energy values are initially not equal to 0, they are used and the value stored in the file is not taken into account.

For operation, "Multi-counter" energy meters have the *LoadData* and *SaveData* functions. While the *LoadData* function overwrites the actual values with stored values, the *SaveData* function saves the current values on the file on the memory card (only one data set is saved). The data is not saved automatically but must be explicitly saved by running *SaveData* in the application program. A power failure is not covered by this function. In this case, there is the option of creating the run variables in the retain memory.

In addition to simple energy metering, "multi-counter" energy meters provide the following additional power analyses:

### Min./max. power

Measurement and output of minimum and maximum power value (absolute). The two values are determined only if the energy meter is running at the same time (the *xRunning* variable has the value "TRUE"). For example, if the power is between -5 kW and 7 kW, the -5 kW is output as the minimum value and the 7 kW as the maximum value. The determined values are retained until the next recording, initialization or reset.

### 15 min. mean value

The 15 min. mean power value is determined and output. The 15 min. mean power value is a variable that can affect the electricity price.

The following method is used to limit the number of measured values (e.g. with a bus cycle of 1 ms = 900000 values):

- For example, the power values available in a 1 ms, 5 ms or 10 ms cycle are added to a variable and saved to a ring buffer once per second. The buffer holds 60 values, i.e. 60 s.
- The contents of the second ring buffer is totaled once per minute and divided by 60. This minute value is saved to the 15 min. ring buffer.
- The 15 min. ring buffer is analyzed every minute, at which time all 15 values are totaled and divided by 15. The result is then output and is current for one minute.

## EnergyMultiCounter

Variables for handling the energy meter, i.e. to activate and deactivate the function and to output status information and energy values.

Variable name	Description
xEnable	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Start energy meter</li> <li>• FALSE – Stop energy meter</li> </ul>

Variable name	Description
xReset	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Reset messages</li> <li>• FALSE – Do not reset messages</li> </ul>
xError	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Error present</li> <li>• FALSE – No error present</li> </ul>
xWarning	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Warning present</li> <li>• FALSE – No warning present</li> </ul>
udiMessageID	Data type: UDINT
	Message ID number
sAdditionalText	Data type: STRING
	Additional message text
xGetAccessControl	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Request access</li> <li>• FALSE – Return access</li> </ul>
xControlActive	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Access granted</li> <li>• FALSE – Access denied</li> </ul>
xResetCounter	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE - Reset counter reading.</li> <li>• FALSE - Do not reset counter reading.</li> </ul>
rEAct_Cmpl_InWs	Data type – REAL
	Sum of all energy values in [W]
rEAct_Neg_InWs	Data type – REAL
	Sum of all negative energy values in [W]
rEAct_Pos_InWs	Data type – REAL
	Sum of all positive energy values in [W]
r15MinAverage	Data type – REAL
	Mean power value of the last 15 min in [W]
MaxPwr	Data type – REAL
	Maximum value of the applied power in [W]
MinPwr	Data type – REAL
	Minimum value of the applied power in [W]

*EnergyMultiIntervalCounter*

The *EnergyMultiIntervalCounter* multi-counter contains the functionality of the *EnergyMultiCounter* multi-counter as well as the interval function of the *EnergyIntervalCounter* single counter. For information on using the interval function and the available operating modes, see chapter "EnergyIntervalCounter" (→ 26).

The 15 min. power monitor does not fit into the system and keeps running regardless. Current power values are available irrespective of the operating mode.

Variables for handling the energy meter, i.e. to activate and deactivate the function and to output status information and energy values.

Variable name	Description
xEnable	Data type – BOOL
	<ul style="list-style-type: none"> <li>TRUE – Start energy meter</li> <li>FALSE – Stop energy meter</li> </ul>
xReset	Data type – BOOL
	<ul style="list-style-type: none"> <li>TRUE – Reset messages</li> <li>FALSE – Do not reset messages</li> </ul>
xError	Data type – BOOL
	<ul style="list-style-type: none"> <li>TRUE – Error present</li> <li>FALSE – No error present</li> </ul>
xWarning	Data type – BOOL
	<ul style="list-style-type: none"> <li>TRUE – Warning present</li> <li>FALSE – No warning present</li> </ul>
udiMessageID	Data type: UDINT
	Message ID number
sAdditionalText	Data type: STRING
	Additional message text
xGetAccessControl	Data type – BOOL
	<ul style="list-style-type: none"> <li>TRUE – Request access</li> <li>FALSE – Return access</li> </ul>
xControlActive	Data type – BOOL
	<ul style="list-style-type: none"> <li>TRUE – Access granted</li> <li>FALSE – Access denied</li> </ul>
xResetCounter	Data type – BOOL
	<ul style="list-style-type: none"> <li>TRUE - Reset counter reading.</li> <li>FALSE - Do not reset counter reading.</li> </ul>
rAct_InWs	Data type – REAL
	Current energy value in [W]
xRunning	Data type – BOOL
	<ul style="list-style-type: none"> <li>TRUE – Recording in progress</li> <li>FALSE – Recording not in progress</li> </ul>

Variable name	Description
xSingleShot	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Activate "single measurement" operating mode.</li> <li>• FALSE – Deactivate "single measurement" operating mode.</li> </ul>
xStartStop	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Activate "normal" operating mode. (Similar to the function of the storage unit oscilloscope)</li> <li>• FALSE – Deactivate "normal" operating mode.</li> </ul>
xTrigger	Data type – BOOL
	<ul style="list-style-type: none"> <li>• TRUE – Activate triggering at positive edge</li> <li>• FALSE – Do not activate triggering</li> </ul>
rEAct_Cmpl_InWs	Data type – REAL
	Sum of all energy values in [W]
rEAct_Neg_InWs	Data type – REAL
	Sum of all negative energy values in [W]
rEAct_Pos_InWs	Data type – REAL
	Sum of all positive energy values in [W]
r15MinAverage	Data type – REAL
	Mean power value of the last 15 min in [W]
MaxPwr	Data type – REAL
	Maximum value of the applied power in [W]
MinPwr	Data type – REAL
	Minimum value of the applied power in [W]

## 6.3.3 SEW PES RealTimeScope

## INFORMATION



The function block is not integrated via automatic code generation but must be implemented manually in the project.

The *SEW PES RealTimeScope* library provides a real-time oscilloscope that can log any variables of the application program with the data type REAL.

## Operating principle

Data is recorded on the MOVI-C® CONTROLLER in single-shot mode using the *RealTimeScope* function block. The data is saved in a CSV file on a Windows system (e.g. Windows part of the MOVI-C® CONTROLLER or external PCs). For storage, the function block provides a buffer variable that is used to exchange one line of the CSV file at a time. Also included are other variables for the handshake and progress information to control the transfer. Data is exchanged via a standardized communication protocol such as CODESYS ARTI.

To log data, the application program must provide the *RealTimeScope* function block with a ring buffer in the form of a static array. The shortest possible sampling interval is the cycle time of the *HighPrio* task of the application program, but can also be any integer multiple of the cycle time. Since the size of REAL variables is 4 bytes, the buffer size in bytes must be an integer divisible by 4. On the one hand, the necessary buffer size depends on the sampling interval, the desired recording duration and the number of channels to be recorded (max. 8); on the other hand, it should be selected so that it only occupies as much storage as absolutely necessary.

The required buffer size in bytes is calculated as follows:

$$\text{Buffer size[Byte]} = \left( \frac{\text{Measurement duration[ms]}}{\text{Sampling interval[ms]}} + 1 \right) \cdot 4 \cdot \text{Number of channels}$$

28633552651

For example, the following buffer size is required for 4-channel recording with a sampling interval of 1 ms and a recording time of 5 s:

$$\text{Buffer size[Byte]} = \left( \frac{5000[\text{ms}]}{1[\text{ms}]} + 1 \right) \cdot 4 \cdot 4 = 80.016 \approx 78 \text{ KB}$$

28633555851

Various variables are provided in the *Measurement* structure for transferring the CSV file from the MOVI-C® CONTROLLER to the connected Windows system. See chapter "Measurement" (→ 37).

If a recording is successful, a CSV file is provided for transfer. The transfer takes place line by line. In each cycle, the *RealTimeScope* function block first checks the status of the variable *INOUT.xNewData*. If this variable has the value "FALSE", the receiver side is ready. In this case, one line of the CSV file is written to the variable *OUT.abbyLineBuffer* and the variables *OUT.diLines2Transfer* and *OUT.diLinesTransferred* are set to the corresponding values. The variable *INOUT.xNewData* is then set to "TRUE" to signal to the receiver side that a new row is ready for transfer. As soon as the receiver side has fetched the line, it must set the variable *INOUT.xNewData* to "FALSE" to signal to the function block that the next line can be transferred. This procedure is repeated until all lines of the file have been successfully transferred.



## Integration

Do the following to integrate the functionality:

1. Create an instance of the *RealTimeScope* function block in the *User\_PRG* program and provide a ring buffer for logging:

```
⇒ FbScope:SEW_RealTimeScope.RealTimeScope;
   xInitDoneScope:BOOL;
   arScopeBuffer:ARRAY[0..524287] OF REAL
```

2. In the *Init* action in the *User\_PRG* program, initialize the *RealTimeScope* function block and assign the recording channels as well as the basic settings for the measurement. The following code block shows an example of the initialization of the *RealTimeScope* function block in the *Init* action of the *User\_PRG* program as well as the assignment of the recording channels and the basic settings for the measurement:

```
⇒ //Initialization of function block
   IF NOT xInitDone THEN
     xInitDoneScope:=FbScope.Init(
       prBuffer:=ADR(arScopeBuffer),
       diBufferSize:=ANY_TO_DINT(SIZEOF(arScopeBuffer)),
       diCycleTime:=ANY_TO_DINT(gc_uiTaskCycleTime));

     //Default settings of the scope measurement
     //Channel assignment
     FbScope.MapChannel(0,ADR(rVltgDcLink),'Volt.DC-Link [V]');
     FbScope.MapChannel(1,ADR(rPwrMDP),'Power MDP [kW]');
     FbScope.MapChannel(2,ADR(rPwrAxis1),'Power Axis1 [kW]');
     FbScope.MapChannel(3,ADR(rVelAxis1),'Veloc. Axis1 [rpm]');
     //Label in the time column in the .csv file
     FbScope.SetTimeCaption('Time[ms]');
     //Activation of the measurement channels
     FbScope.stMeasurement.IN.axEnable[0]:=TRUE;
     FbScope.stMeasurement.IN.axEnable[1]:=TRUE;
     FbScope.stMeasurement.IN.axEnable[2]:=TRUE;
     FbScope.stMeasurement.IN.axEnable[3]:=TRUE;
     FbScope.stMeasurement.IN.axEnable[4]:=FALSE;
     FbScope.stMeasurement.IN.axEnable[5]:=FALSE;
     FbScope.stMeasurement.IN.axEnable[6]:=FALSE;
     FbScope.stMeasurement.IN.axEnable[7]:=FALSE;
     //Trigger settings
     FbScope.stMeasurement.IN.eTrigSource:=
       SEW_RealTimeScope.E_Scope_TrigSource.Channel;
     FbScope.stMeasurement.IN.diTrigChannel:=DINT#0;
     FbScope.stMeasurement.IN.eTrigMode:=
       SEW_RealTimeScope.E_Scope_TrigMode.RisingEdge;
     FbScope.stMeasurement.IN.rTrigPosition:=0.25;
     FbScope.stMeasurement.IN.rTrigLevel:=500;
     //Time base settings
     FbScope.stMeasurement.IN.diMeasureTime:=DINT#5000;
     FbScope.stMeasurement.IN.diSampleTime:=DINT#1;

     //Default settings of the control inputs
     FbScope.stMeasurement.IN.xReset:=FALSE;
     FbScope.stMeasurement.IN.xStart:=FALSE;
```

```

FbScope.stMeasurement.IN.xStop:=FALSE;

//Name of the .csv file
FbScope.stMeasurement.IN.sFilename:='ScopeData.csv';

//Result of the Init action of the User PRG
xInitDone:=xInitDoneScope;
END_IF

```

3. In the *Main* action of the *User\_PRG* program, call the *CallMain* method of the instance of the *RealTimeScope* function block:

⇒ `FbScope.CallMain();`

4. In the *HighPrio* action of the *User\_PRG* program, call the *CallHighPrio* method of the instance of the *RealTimeScope* function block:

⇒ `FbScope.CallHighPrio();`

- ⇒ When you log on to the controller, you can now display the variables by double-clicking the *User\_PRG* program in the device tree under the instance of the *RealTimeScope* function block:

Measurement	ST_Scope_Measurement	
IN	ST_Scope_Measurement_IN	
axEnable	ARRAY [0..(GVL.gc_diMaxChannels - 1)] OF BOOL	
axEnable[0]	BOOL	TRUE
axEnable[1]	BOOL	TRUE
axEnable[2]	BOOL	TRUE
axEnable[3]	BOOL	TRUE
axEnable[4]	BOOL	FALSE
axEnable[5]	BOOL	FALSE
axEnable[6]	BOOL	FALSE
axEnable[7]	BOOL	FALSE
diMeasureTime	DINT	60000
diSampleTime	DINT	1
eTrigMode	E_SCOPE_TRIGMODE	Immediate
rTrigLevel	REAL	0
rTrigPosition	REAL	0
eTrigSource	E_SCOPE_TRIGSOURCE	Channel
diTrigChannel	DINT	0
xStart	BOOL	FALSE
xStop	BOOL	FALSE
sFilename	STRING(255)	'ScopeData.csv'
xKeepIncompleteFile	BOOL	FALSE
OUT	ST_Scope_Measurement_OUT	
xWaiting	BOOL	FALSE
xRecording	BOOL	FALSE
xSaving	BOOL	FALSE
diLines2Write	DINT	0
diLinesWritten	DINT	0
xDone	BOOL	FALSE
xAborted	BOOL	FALSE

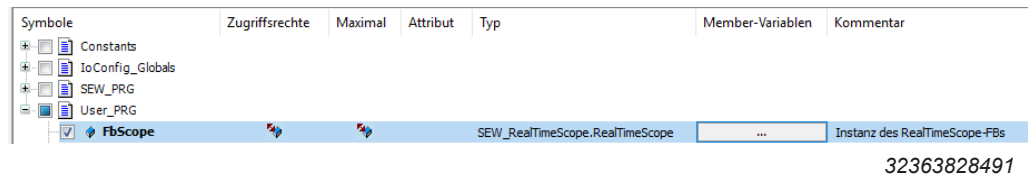
31193411723

26625539/EN – 03/2020

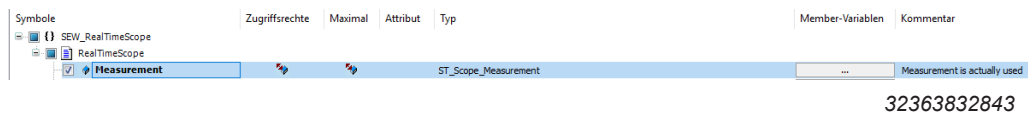
## Symbol configuration

Non-volatile data recorded with the function block must be stored externally. A Windows system is required for this purpose, which is either located on the Windows part of the MOVI-C® CONTROLLER (if available) or on a PC connected to the MOVI-C® CONTROLLER via the engineering interface. To transfer the recorded CSV file to the Windows system via CODESYS ARTI, the required variables must be made known to the Windows system. Proceed as follows:

1. Create a symbol configuration in the application program.
2. Specify the instance of the function block in the symbol configuration selection window.



3. Click on the button [...] in the "Member variables" column.  
⇒ A dialog box for selecting the function block member variables to be exported is opened.
4. Select the function block and confirm your selection with [OK].



5. Click on [Generate Code] under [Create] on the menu bar.  
⇒ The symbol configuration is exported as an XML file.
6. Import the generated XML file into the application on the Windows system that retrieves and saves the lines of the CSV file from the MOVI-C® CONTROLLER.

## Methods

### Init

Method for initializing the block. The method must be called in the *Init* action of the *User\_PRG* program.

Variable name	Description
prBuffer	Data type – POINTER TO REAL Pointer to the ring buffer for recording measurement data Information: Since the measurement data is recorded in REAL format (equal to 4 bytes per value), the buffer size must be an integer divisible by 4.
diBufferSize	Data type - DINT Size of the ring buffer in bytes
diCycleTime	Data type - DINT Cycle time of the <i>HighPrio</i> task of the program in [ms]

*CallMain*

Method for running the program components of the block that are to be processed cyclically in the free-running task of the application program. The method must be called in the *Main* action of the *User\_PRG* program.

*CallHighPrio*

Method for running the program components of the block that are to be processed cyclically in the real-time task of the application program. The method must be called in the *HighPrio* action of the *User\_PRG* program.

*MapChannel*

Method for assigning the channels of the block to the variables to be recorded.

Variable name	Description
diChannel	Data type - DINT
	Number of the channel to which a variable is to be assigned.
prVariable	Data type – POINTER TO REAL
	Pointer to the variable to which the channel is to be assigned.
sCaption	Data type: STRING(127)
	Identifier of the signal for the header of the .CSV file to be saved

*MapExtTrig*

Method for assigning a variable that will serve as a trigger signal for the external trigger input.

Variable name	Description
prExtTrig	Data type – POINTER TO REAL
	Pointer to the variable that will serve as the trigger signal.

*SetTimeCaption*

Method for assigning an identifier of the time column to the header of the .CSV file to be saved.

Variable name	Description
sTimeCaption	Data type: STRING(127)
	Identifier of the time column for the header of the .CSV file to be saved

**Diagnostics**

The variables available in this structure are described in chapter "Diagnostics" (→ 17).

## Measurement

Structure for operating the *RealTimeScope* function block, i.e., for starting and stopping measurements, specifying the name of the .CSV file to be saved, monitoring the recording status, etc.

IN

Variable name	Description
axEnable	Data type – ARRAY [0..7] OF BOOL  Individually specify whether the data of channels 0 to 7 is to be recorded. The following applies to each channel: <ul style="list-style-type: none"> <li>• TRUE – Record data of the channel</li> <li>• FALSE – Do not record data of the channel</li> </ul> The prerequisite for recording the data of a channel is that it has been previously assigned correctly using the <i>MapChannel()</i> method.
diMeasureTime	Data type - DINT  Recording time in [ms]
diSampleTime	Data type - DINT  Sampling time in [ms]  (At minimum, the cycle time of the <i>HighPrio</i> task of the application)
eTrigMode	Data type – ENUMERATION  Selecting the trigger mode: <ul style="list-style-type: none"> <li>• Immediate – Measurement starts immediately <i>xStart</i> has rising edge</li> <li>• RisingEdge – Measurement starts immediately if trigger signal has rising edge</li> <li>• FallingEdge – Measurement starts immediately if trigger signal has falling edge</li> </ul>
rTrigLevel	Data type – REAL – floating-point number  Trigger level for edge detection
rTrigPosition	Data type – REAL – floating-point number  Position of the trigger time on the time axis as a fraction of the measurement time, for example: <ul style="list-style-type: none"> <li>• 0.0 – Start of time axis</li> <li>• 0.5 – Middle of time axis</li> <li>• 1.0 – End of time axis</li> </ul>

Variable name	Description
eTrigSource	Data type – ENUMERATION Selection of the trigger source: <ul style="list-style-type: none"> <li>Channel – Triggering in response to the measurement signal of a channel</li> <li>External – Triggering in response to external signal</li> </ul> The prerequisite for external triggering is that the external trigger signal was previously assigned correctly using the <i>MapExtTrig()</i> method.
diTrigChannel	Data type - DINT Number of the channel acting as the trigger source (if <i>eTrigSource</i> is set to Channel)
xStart	Data type – BOOL <ul style="list-style-type: none"> <li>TRUE – Start measurement or activate trigger</li> <li>FALSE – Do not start measurement</li> </ul>
xStop	Data type – BOOL <ul style="list-style-type: none"> <li>TRUE – Stop measurement or saving of recording</li> <li>FALSE – Do not stop measurement</li> </ul>
sFileName	Data type: STRING(255) Name of the .CSV file to be logged

## OUT

Variable name	Description
xWaiting	Data type – BOOL <ul style="list-style-type: none"> <li>TRUE – Waiting for a trigger event</li> <li>FALSE – Not waiting for a trigger event</li> </ul>
xRecording	Data type – BOOL <ul style="list-style-type: none"> <li>TRUE – Recording in progress</li> <li>FALSE – Recording not in progress</li> </ul>
xTransferring	Data type – BOOL <ul style="list-style-type: none"> <li>TRUE - The recording data is transferred.</li> <li>FALSE - No transfer active.</li> </ul>
diLines2Transfer	Data type - DINT Number of CSV file lines to be transferred
diLinesTransferred	Data type - DINT Number of CSV file lines already transferred
xDone	Data type – BOOL <ul style="list-style-type: none"> <li>TRUE - Measurement and transfer of the data are completed.</li> <li>FALSE - Measurement and transfer of the data are not yet completed.</li> </ul>

Variable name	Description
xAborted	Data type – BOOL
	<ul style="list-style-type: none"><li>• TRUE - Measurement or transfer of data interrupted by the user.</li><li>• FALSE - No interruption of the measurement or transfer of data by the user.</li></ul>
xError	Data type – BOOL
	<ul style="list-style-type: none"><li>• TRUE – Software module has a fault status</li><li>• FALSE – Software module does not have a fault status</li></ul>
abyLineBuffer	Data type - ARRAY [0..GVL.gc_diLineBufSize-1] OF BYTE
	Buffer that provides the current line of the CSV file to be transferred

#### INOUT

Variable name	Description
xNewData	Data type – BOOL
	Handshake variable for transferring the CSV file. <ul style="list-style-type: none"><li>• TRUE - The function block has provided a new line for transfer in <i>abyLineBuffer</i>.</li><li>• FALSE - No transfer active. The function block can provide a new line in <i>abyLineBuffer</i> for transfer.</li></ul>

## 7 Fault management

### 7.1 Fault codes

#### 7.1.1 RealTimeScope

16#9080	The pointer to the scope data buffer is a null pointer or the buffer size is 0.
16#9081	The cycle time is below the permissible minimum cycle time (1 ms).
16#9082	The sampling interval is not an integer multiple of the cycle time.
16#9083	The recording time is not an integer multiple of the sampling interval.
16#9084	No channels assigned
16#9085	The trigger position is outside the permissible limits (< 0.0 or > 1.0).
16#9086	Invalid trigger channel number (unused or deactivated channel).
16#9087	The external trigger input is not properly assigned (zero pointer).
16#9088	The amount of data to be recorded exceeds the buffer size.
16#9089	The transferred file name is empty.
16#908A	The length of the last edited line exceeds the size of the line buffer.
16#908B	Failure to connect to the license manager during initialization.
16#908C	Failure to connect to the measurement structure during initialization.
16#908D	Failure to call internal initialization routines during initialization.

#### 7.1.2 Data logger

16#90A0	The cycle time is below the minimum cycle time (1 s).
16#90A1	The time interval between two measurements is not an integer multiple of the cycle time.
16#90A2	No channels assigned
16#90A3	The length of the last edited line exceeds the size of the line buffer.
16#90A4	Failure to connect to the license manager during initialization.
16#90A5	During initialization, the connection of the logging structure failed.
16#90A6	Failure to call internal initialization routines during initialization.



## Index

### A

Access rights ..... 18

### C

Concurrent access ..... 18

Copyright notice ..... 6

### D

Decimal separator ..... 6

### E

Embedded safety notes ..... 6

EtherCAT®  
Beckhoff trademark ..... 6

### F

Fault management ..... 40

### H

Hazard symbols  
Meaning ..... 6

### L

Licensing ..... 12

### M

Monitor access ..... 18

### N

Notes  
Designation in the documentation ..... 5

Meaning of the hazard symbols ..... 6

### P

Product names ..... 6

Project planning ..... 12

### R

Rights to claim under limited warranty ..... 6

### S

Safety notes

Bus systems ..... 8

Designation in the documentation ..... 5

Meaning of the hazard symbols ..... 6

Preliminary information ..... 8

Structure of embedded ..... 6

Structure of section-related ..... 5

Section-related safety notes ..... 5

Short designation ..... 7

Signal words in safety notes ..... 5

Startup ..... 13

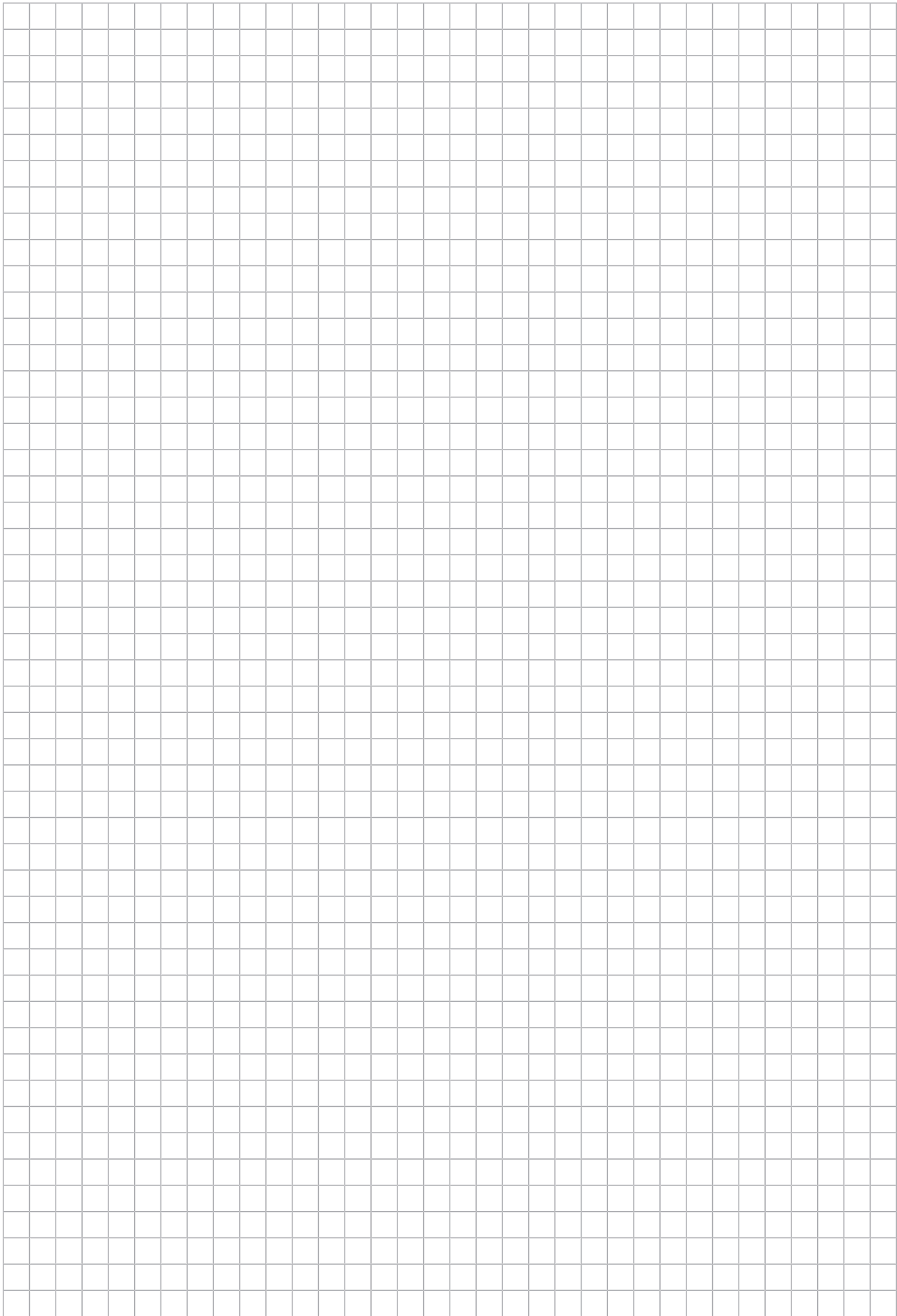
### T

Target group ..... 8

Trademarks ..... 6

### U

Use, designated ..... 8







**SEW-EURODRIVE**  
Driving the world

**SEW**  
**EURODRIVE**

SEW-EURODRIVE GmbH & Co KG  
Ernst-Blickle-Str. 42  
76646 BRUCHSAL  
GERMANY  
Tel. +49 7251 75-0  
Fax +49 7251 75-1970  
sew@sew-eurodrive.com  
→ [www.sew-eurodrive.com](http://www.sew-eurodrive.com)