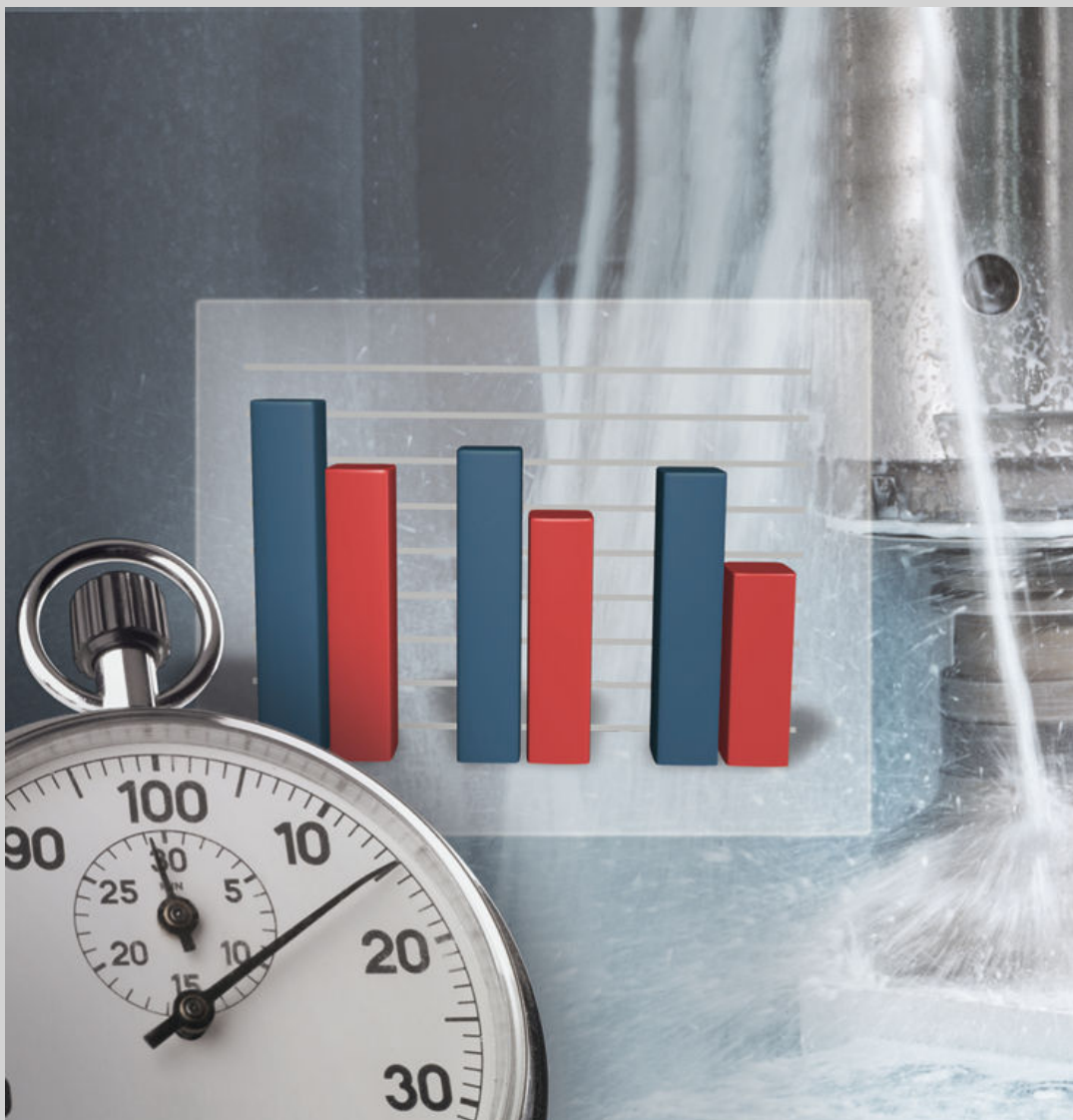


IndraWorks 14VRS

Energy Efficiency Management

Application Description
R911339229

Edition 06



Title	IndraWorks 14VRS Energy Efficiency Management
Type of Documentation	Application Description
Document Typecode	DOK-IWORKS-4EE*****V14-AP06-EN-P
Internal File Reference	RS-d3607a8fc780d3a80a6846a500b8005f-6-en-US-3
Change Record	Edition 06, 2018-04 Refer to tab. 1-1 "Change Record" on page 5
Copyright	© Bosch Rexroth AG 2018 All rights reserved, also regarding any disposal, exploitation, reproduction, editing, distribution, as well as in the event of applications for industrial property rights.
Liability	The specified data is intended for product description purposes only and shall not be deemed to be a guaranteed characteristic unless expressly stipulated in the contract. All rights are reserved with respect to the content of this documentation and the availability of the product.
Editorial Department	System IndraLogic & Basis Motion Logic, MiAI (TaDo/MePe)

Table of Contents

	Page
1 About this documentation.....	5
1.1 Validity of the documentation.....	5
1.2 Documentation structure.....	6
1.3 Using safety instructions.....	6
1.3.1 Structure of the safety instructions.....	6
1.3.2 Explaining signal words and safety alert symbol.....	6
1.3.3 Symbols used.....	8
1.3.4 Explaining the signal alert symbol on the device.....	8
1.4 Required and supplementing documentation XLC/MLC.....	8
1.5 Required and supplementing documentation MTX.....	14
1.5.1 Selection/compilation.....	14
1.5.2 Configuration.....	14
1.5.3 Commissioning.....	15
1.5.4 Operation.....	16
1.5.5 OEM Engineering.....	17
1.5.6 AddOns.....	17
1.6 Names and abbreviations.....	17
1.7 Customer feedback.....	18
2 Introduction.....	19
3 EnergyManagement.....	21
3.1 Overview.....	21
3.2 EnergyMonitoring.....	23
3.2.1 Motivation.....	23
3.2.2 Next steps.....	23
3.2.3 Functionality.....	24
3.2.4 Architecture.....	24
3.2.5 Functioning.....	25
3.3 EnergyControl.....	25
3.3.1 Motivation.....	25
3.3.2 Next steps.....	26
3.3.3 Sercos Energy.....	26
3.3.4 Functionality.....	27
3.3.5 Architecture.....	27
3.3.6 Functioning.....	28
3.3.7 Modularization.....	28
3.3.8 Coupling.....	29
4 Commissioning example: Handling and press.....	31
4.1 Overview.....	31
4.2 Module: Handling.....	31
4.3 Module: Press.....	32

Table of Contents

	Page
5	Configuring the EnergyManagement function blocks..... 35
5.1	Overview..... 35
5.2	EnergyMonitoring..... 35
5.2.1	Overview..... 35
5.2.2	Handling and press..... 35
5.3	EnergyControl..... 36
5.3.1	Setting the energy modes for machine and stations..... 36
5.3.2	Safety..... 37
5.3.3	Specifications and commands..... 37
5.3.4	Handling..... 37
5.3.5	Press..... 42
6	Realizing the EnergyMonitoring..... 49
6.1	Overview..... 49
6.2	Declarations and nesting of the function blocks..... 49
6.3	Configuring the EnergyMonitoring..... 50
6.3.1	Overview..... 50
6.3.2	Configuring the EnergyMonitor block..... 50
6.3.3	Configuring the HMIDataCollector block..... 50
6.3.4	Configuring the MonitoringDevice blocks..... 50
6.4	Connecting the consumers to EnergyMonitoring..... 51
6.5	Using commissioning visualization..... 52
7	Realizing EnergyControl..... 55
7.1	Overview..... 55
7.2	Declarations and nesting of the function blocks..... 55
7.3	Configuring the EnergyControl..... 56
7.3.1	Overview..... 56
7.3.2	Configuring the EnergyControl function blocks..... 56
7.3.3	Configuring the command function blocks..... 61
7.3.4	Configuring the BasicDevice function blocks..... 65
7.4	Connecting the consumers to EnergyManagement..... 68
7.4.1	Overview..... 68
7.4.2	Connecting conveyor belts and lighting..... 69
7.4.3	Connecting axes for MLC..... 69
7.4.4	Connecting the axes for MTX..... 70
7.4.5	Initializing the consumers..... 71
7.4.6	Switching consumers after initialization..... 73
7.5	Using commissioning visualization..... 74
7.6	Visualizing "SmartAutomation"..... 76
8	HMI Controls..... 79
8.1	IndraWorks Operation Desktop..... 79
8.1.1	EnergyMonitor..... 79
8.1.2	PowerMonitor..... 81

Table of Contents

	Page
8.1.3 EnergyManagement.....	85
8.2 WinStudio.....	88
8.2.1 Overview.....	88
8.2.2 Inserting .NET controls in WinStudio.....	89
8.2.3 Control interfaces and operation.....	91
8.3 Web-based visualization.....	98
8.3.1 Overview.....	98
8.3.2 EnergyMonitoring.....	99
9 RIL_4EEEnergyManagement.library.....	111
9.1 Overview.....	111
9.2 EnergyMonitoring.....	111
9.2.1 Overview.....	111
9.2.2 Function blocks and their methods.....	113
9.2.3 Data types.....	194
9.2.4 HMI control data types.....	201
9.3 EnergyControl.....	210
9.3.1 Overview.....	210
9.3.2 Function Blocks and Their Methods.....	212
9.3.3 Data types.....	263
9.3.4 HMI control data types.....	269
9.4 PROFIenergy.....	276
9.4.1 Function blocks.....	276
9.4.2 Data types.....	287
9.5 Data types for the MES/control system interface.....	290
9.5.1 MachineEnergyObjectType.....	290
9.5.2 MachineEnergyMonType.....	291
9.5.3 MachineEnergyMonConsumptionType.....	292
9.5.4 MachineEnergyMonRingbufferType.....	293
9.5.5 MachineEnergyMonDevicesType.....	293
9.5.6 MachineEnergyCtrlType.....	293
9.5.7 MachineEnergyCtrlConfigType.....	294
9.5.8 MachineEnergyCtrlStatusType.....	295
9.5.9 MachineEnergyCtrlCmdControlType.....	295
9.5.10 MachineEnergyCtrlCmdStatusType.....	296
9.5.11 IL_4EE_RINGBUFFER_MODE.....	296
9.5.12 IL_4EE_ENERGY_CONTROL_STATE.....	296
9.6 Parameter list.....	297
10 Service and support.....	299
Index.....	301

1 About this documentation

Editions of this documentation

Edition	Release Date	Notes
01	2014-05	First edition MLC/XLC 14VRS, incl. LibraryRIL_4EEEnergyManagement
02	2014-11	New function blocks added
03	2015-08	New function blocks and structures added
04	2015-10	New function blocks and structures added, new chapter "Web-based visualization"
05	2016-01	New function blocks and structures added, new chapter "Web-based visualization"
06	2018-04	New function blocks: • IL_4EELoadProfileEstimation • IL_4EEMonCompAirDeviceType01 • IL_4EEMonFluidDeviceType01 • IL_4EEMonGasDeviceType01 • IL_4EEMonGasDeviceType02 • IL_4EEMonPneumaticDeviceType01 • IL_4EEMonS0DeviceType01

Tab. 1-1: Change Record

1.1 Validity of the documentation

This documentation is intended for users wanting to optimizing machines and systems with regard to energy consumption.

This documentation is used to describe the Energy Management Systems (4EE):

- Interface
- HMI dialogs
- Commissioning
- PLC libraries

Target group This documentation is intended for programming engineers, commissioning engineers and machine operators of a system.

Required qualification: Individual who is able to assess the tasks assigned and to identify possible safety risks owing to qualification in the subject, knowledge and experience. The individual should also be familiar with the standards and regulations.

Application phases This documentation supports the user during the following phases:

- Engineering
- Commissioning
- Operating and programming

About this documentation

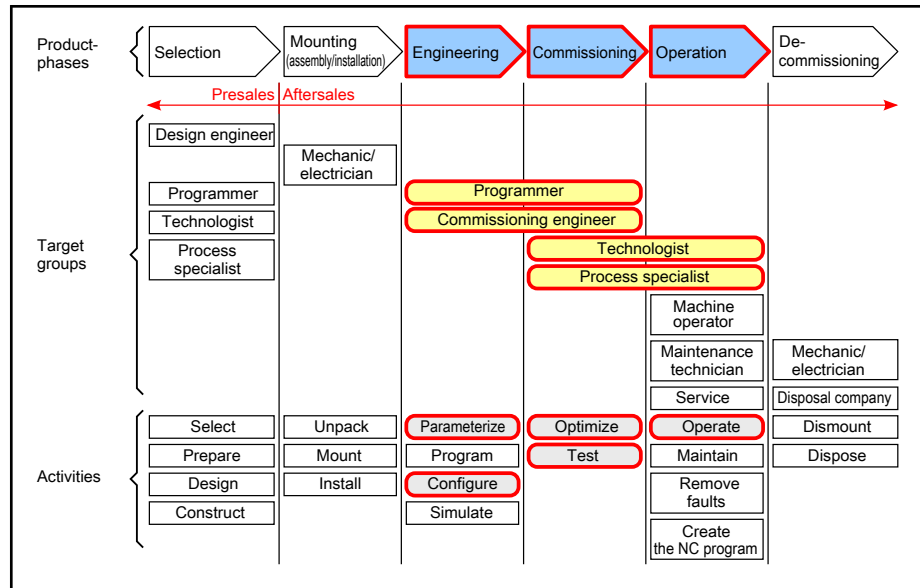


Fig. 1-1: Application phases

1.2 Documentation structure

The [chapter 3 "EnergyManagement" on page 21](#) contains introductory explanation about the energy management issue.

From [chapter 4 "Commissioning example: Handling and press" on page 31](#), a commissioning example is described.

[chapter 8 "HMI Controls" on page 79](#) describes the HMI interfaces Operation Desktop and WinStudio.

[chapter 9 "RIL_4EEEnergyManagement.library" on page 111](#) describes the function blocks and data types of the IL_4EE PLC library and its effects.

1.3 Using safety instructions

1.3.1 Structure of the safety instructions

The safety instructions are structured as follows:

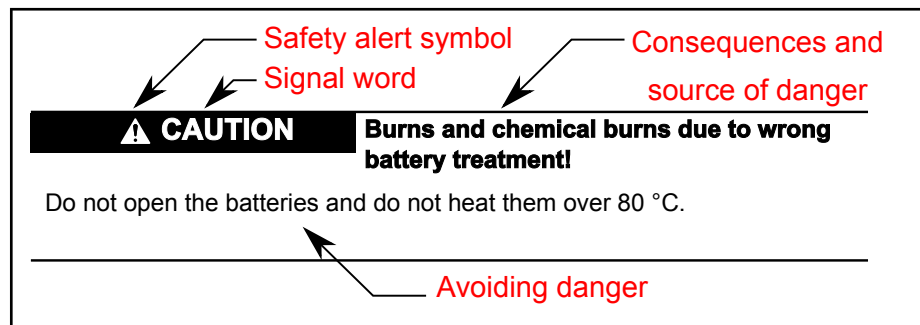


Fig. 1-2: Structure of the safety instructions

1.3.2 Explaining signal words and safety alert symbol

The safety instructions in this documentation contain specific signal words (danger, warning, caution, notice) and, if necessary, a safety alert symbol (according to ANSI Z535.6-2006).

The signal word draws attention to the safety instruction and indicates the risk potential.

About this documentation

The safety alert symbol (triangular safety reflector with exclamation marks), preceding the signal words Danger, Warning, Caution indicates hazards for persons.

 DANGER

In case of non-compliance with this safety instruction, death or serious injury **will** occur.

 WARNING

In case of non-compliance with this safety instruction, death or serious injury **can** occur.

 CAUTION

In case of non-compliance with this safety instruction, minor or moderate injury can occur.

NOTICE

In case of non-compliance with this safety instruction, material damage can occur.

About this documentation

1.3.3 Symbols used

Pointers are displayed as follows:



This is a note.

Tips are displayed as follows:



This is a tip.

1.3.4 Explaining the signal alert symbol on the device



If this symbol is on your device, you have to observe the documentation on the device. The respective documentation informs on the type of hazard as well as the steps required to avoid this hazard.

1.4 Required and supplementing documentation XLC/MLC

Documentation titles with type codes and part numbers

System overview
IndraLogic XLC IndraMotion MLC 14VRS System Overview DOK-XLCMLC-SYSTEM**V14-PRRS-EN-P, R911341504 This documentation provides an overview on the possible hardware/software components of the automation systems IndraLogic XLC and IndraMotion MLC of the mentioned version. The documentation facilitates assembling a system.

Tab. 1-2: XCL/MLC documentation overview - System overview

First steps
IndraMotion MLC 14VRS First Steps DOK-MLC***-F*STEP**V14-CORS-EN-P, R911341517 This documentation describes the first steps of the IndraMotion MLC and the RobotControl. It includes the hardware and software prerequisites as well as the creation of a project.
IndraLogic XLC 14VRS First Steps DOK-XLC***-F*STEP**V14-CORS-EN-P, R911341489 This documentation describes the first steps of the IndraLogic XLC. It includes the hardware and software prerequisites as well as the creation of a project.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Project Conversion DOK-XLCMLC-PROCONV*V14-APRS-EN-P, R911341494 This documentation describes the project conversion of the IndraMotion MLC.

Tab. 1-3: XCL/MLC documentation overview - First steps

Engineering
IndraWorks 14VRS Software Installation DOK-IWORKS-SOFTINS*V14-CORS-EN-P, R911344286 This documentation describes the IndraWorks installation.
IndraWorks 14VRS Engineering DOK-IWORKS-ENGINEE*V14-APRS-EN-P, R911343566 This documentation describes the application of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.
IndraLogic XLC IndraMotion MLC 14VRS Function Description DOK-XLCMLC-FUNC***V14-APRS-DE-P, R911341700 This documentation describes wizards, context menus, dialogs, control commissioning, device configuration and functionalities of the IndraMotion MLC.

Tab. 1-4: XCL/MLC documentation overview - Engineering

Diagnostics and service
IndraLogic XLC IndraMotion MLC 14VRS Diagnostics DOK-XLCMLC-DIAG***V14-RERS-EN-P, R911341481 This documentation includes all control diagnostics implemented in the control systems IndraLogic XLC and IndraMotion MLC.
IndraLogic XLC IndraMotion MLC 14VRS Parameters DOK-XLCMLC-PARAM***V14-RERS-EN-P, R911341479 This documentation describes the parameters of the XLC/MLC systems as well as the interaction between parameterization and programming. It includes the axis parameters, control parameters, kinematic parameters, touch probe parameters and programmable limit switch parameters.
IndraLogic XLC IndraMotion MLC 14VRS Commissioning DOK-XLCMLC-STARTUP*V14-CORS-EN-P, R911341502 This documentation describes the steps to commission and service the IndraMotion MLC and IndraLogic XLC systems. It includes checklists for frequent tasks and a detailed description of the steps.
WebAssistant DOK-GENERL-WEB*ASSIST*-APRS-EN-P, R911381469 This documentation describes the WebAssistant. The WebAssistant is a webbased diagnostic tool used to access a control system via an Ethernet highspeed connection. The WebAssistant allows OEMs, end users and service engineers to access and to remotely diagnose a system. A PC with Internet Explorer version 11 or higher, Firefox version 45 or above or Google Chrome version 54 or above is required. The following control variants are supported: • IndraMotion MLC L25/L45/L65/L75/XM2 • IndraLogic XLC L25/L45/L65/L75/VEP
IndraWorks IndraMotion Service Tool DOK-IWORKS-IMST*****-APRS-DE-P, R911341383 This documentation describes the IndraMotion Service Tool (IMST). IMST is a web-based diagnostic tool used to access a control system via an Ethernet high-speed connection. The IMST allows OEMs, end users and service engineers to access and remotely diagnose a system. The PC has to use at least Internet Explorer 8, Firefox 3.5 or a higher version. The following control variants are supported: • IndraMotion MLC L25/L45/L65/L75/XM12/XM2/VPx • IndraLogic XLC L25/L45/L65/L75

Tab. 1-5: XCL/MLC documentation overview - Diagnostics and service

About this documentation

Firmware
Rexroth IndraDrive MPx-20, Functions ¹⁾ DOK-INDRV*-MP*-20VRS**-AP01-EN-P, R911345608 This documentation describes all functional properties of the IndraDrive firmware in the variants MPB-20, MPM-20, MPC-20 and MPE-20.
Rexroth IndraDrive MPx-16 to MPx-20 und PSB, Parameters DOK-INDRV*-GEN1-PARA**-RERS-EN-P, R911328651 This documentation contains the descriptions of all parameters implemented in the firmware for drive controllers of the IndraDrive range. It supports the drive controller parameterization.
Rexroth IndraDrive MPx-16 to MPx-20 and PSB, Diagnostic Messages DOK-INDRV*-GEN1-DIAG**-RERS-EN-P, R911326738 This documentation contains the descriptions of all diagnostic messages implemented in the following firmware products: • Drive controller firmware variants MPx-16 to MPx-20 • PSB firmware variants for "HMU05" and "KMV03" type supply units • Firmware variants for "HMF01" type supply units It assists machine operators and installation programmers with troubleshooting.
Hardware
Rexroth IndraDrive Control Sections CSB02, CSE02, CSH02, CDB02 DOK-INDRV*-Cxx02*****-PRRS-EN-P, R911338962
Rexroth IndraDrive Cs Drive Systems with HCS01 DOK-INDRV*-HCS01*****-PRRS-EN-P, R911322210
Rexroth IndraDrive Mi Drive Systems with KCU02, KSM02, KMS02/03, KMV03 DOK-INDRV*-KCU02+KSM02-PRxx-EN-P, R911335703
Rexroth IndraDrive ML Drive Systems with HMU05 DOK-INDRV*-Hxx05*****-PRxx-EN-P, R911344279

1)

In preparation

Tab. 1-6:

Overview of documentations for drive controllers

PLC
IndraWorks 14VRS PLC Programming System IndraLogic 2G DOK-IWORKS-IL2GPRO*V14-APRS-EN-P, R911343571 This documentation describes the PLC programming tool IndraLogic 2G and its usage. It includes the basic use, first steps, visualization, menu items and editors.
IndraWorks 14VRS Basic Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V14-LIRS-EN-P, R911343920 This documentation describes the system-comprehensive PLC libraries.
IndraLogic XLC IndraMotion MLC 14VRS PLCopen Libraries DOK-XLCMLC-FUNLIB**V14-LIRS-EN-P, R911341491 This documentation describes the function blocks, functions and data types of the RIL_CommonTypes, ML_Base and ML_PLCopen libraries for the IndraLogic XLC/IndraMotion MLC. It also includes the error reactions of function blocks.

Tab. 1-7:

XCL/MLC documentation overview - PLC

Field buses
IndraWorks 14VRS Field Buses DOK-IWORKS-FB*****V14-APRS-EN-P, R911341485 This documentation describes the field bus and local periphery connections supported by the IndraLogic XLC, IndraMotion MLC and IndraMotion MTX systems. The focus of this documentation is on the configuration, parameterization, commissioning and the diagnostics of the different periphery connections. This documentation is the basis for the online help.
IndraWorks 14VRS Field Buses Libraries DOK-IWORKS-FB*LIB**V14-LIRS-EN-P, R911343575 This documentation describes the field bus libraries: RIL_ProfibusDP_02, RIL_ProfibusDP Slave, RIL_ProfinetIO, RIL_ProfinetIODevice, RIL_EtherNetIPAdapter, RIL_MappingList, RIL_SERCOSIII, RIL_Inline including their diagnostics and error reactions of the function blocks.
Rexroth IndraWorks 12VRS FDT Container DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398 This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.
Sercos System Manual for I/O Devices DOK-CONTRL-ILS3*****-APxx-EN-P, R911333512 This documentation describes the configuration, parameterization, commissioning and diagnostics of I/O devices with a Sercos interface.

xx Corresponding edition
Tab. 1-8: XCL/MLC documentation overview - Field buses

HMI
IndraWorks 14VRS HMI DOK-IWORKS-HMI*****V14-APRS-EN-P, R911343569 This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.
IndraWorks 14VRS WinStudio 7.4 DOK-IWORKS-WINSTUD*V14-APRS-EN-P, R911341585 This "User Manual and Technical Reference Book" facilitates working with the "Rexroth WinStudio" software for optimal results. This document provides technical information and step-by-step instructions to create web-enabled HMI/SCADA programs.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS HMI Connection DOK-XLCMLC-HMI*****V14-APRS-EN-P, R911341497 This documentation describes the visualization systems supported by the IndraLogic XLC and IndraMotion MLC and their connection.

Tab. 1-9: XCL/MLC documentation overview - HMI

About this documentation

Technology
IndraLogic XLC IndraMotion MLC 14VRS Generic Application Template DOK-XLCMLC-TF*GAT**V14-APRS-EN-P, R911341487 This documentation provides a structured template to the IndraLogic PLC programmer. This template can be used to add and edit the PLC programming code. It includes the template, the template wizard and example applications.
IndraMotion MLC 14VRS Technology Libraries DOK-MLC***-TF*LIB**V14-LIRS-EN-P, R911341511 This documentation describes the function blocks, functions and data types of the "ML_TechInterface.library", "ML_TechMotion.library", "RMB_TechCam.library" and "ML_TechBase.library". It also includes libraries for the winder functionality, register controller functionality and CrossCutter functionality.
IndraWorks 14VRS Energy Efficiency Management DOK-IWORKS-4EE*****V14-APRS-EN-P, R911339229 This documentation describes the HMI connection of the energy management system and the function blocks, functions and data types of the "RIL_4EE " library. The documentation also includes the error reactions of the function blocks.
Rexroth IndraMotion MLC 14VRS RegisterControl (Library) DOK-MLC***-REGI*CO*V14-LIRS-EN-P, R911341509 This documentation describes the inputs and outputs of the individual function blocks and provides notes on their use.
IndraMotion MLC 14VRS RegisterControl (Application Description) DOK-MLC***-REGI*CO*V14-APRS-EN-P, R911341507 This documentation describes the application of the integrated register control for a rotogravure printing machine. The components of the mark stream sensor, the HMI application and the error recovery options are described. This instruction provides information on how to operate the register control, react to errors and query diagnostics. This documentation is written for machine setters and machine operators.
Rexroth IndraMotion MLC 14VRS Winder Function Application DOK-MLC***-TF*WIND*V14-APRS-EN-P, R911341513 This application-related system documentation describes the application of the winder technology functions.
Rexroth IndraWorks CamBuilder 13VRS DOK-IWORKS-CAMBUIL*V13-APRS-EN-P, R911336291 This documentation describes the basic principles and operation of the CamBuilder, the cam editing tool.
IndraMotion MLC 14VRS Robot Control V2 DOK-MLC***-ROCO****V14-RERS-EN-P, R911341588 This documentation provides information about the Robot Control V2. The focus is on PLCopen programming. The program structure, variables, functions, motion statements and the required system parameters are described. This documentation also includes the description of the "ML_Robot" library. Robot Control V2 is supported by a kinematic interface. The components of this interface are described in the "ML_KinTech" library.

Tab. 1-10: XCL/MLC documentation overview - Technology

Open Core Engineering - OCE
IndraLogic XLC IndraMotion MLC 14VRS First Steps MLPI (online help) DOK-XLCMLC-MLPI****V14-CORS-EN-E, R911341033 This document describes the installation and the commissioning of the MLPI interface.
IndraLogic XLC IndraMotion MLC 14VRS Automation Interface (online help) DOK-XLCMLC-AUT*INT*V14-APRS-EN-E, R911341499 This documentation describes the script-based access to IndraWorks project data via the interface of the Automation Interface.

Tab. 1-11: XCL/MLC documentation overview - Open Core Engineering - OCE

SafeLogic
IndraWorks SafeLogic 14VRS First Steps DOK-IWORKS-SL*STEP*V14-CORS-EN-P, R911341520 This documentation describes the initial commissioning of the Safety function module and its external Safety periphery. Taking the example of a project, all required steps to commission a simple Safety application are executed.
IndraLogic XLC IndraMotion MLC 14VRS SafeLogic System Overview DOK-XLCMLC-SL**SYS*V14-PRRS-EN-P, R911341696 This documentation describes the project planning, configuration, creation and commissioning of safety-related devices.
IndraWorks 14VRS SafeLogic - Project Configuration, Version 1.2.0.0. DOK-IWORKS-SL**PRJ*V14-APRS-EN-P, R911341694 This documentation describes the configuration of Safety applications.
IndraControl CFL01.1-F1 Safety Function Module DOK-CTRL-SL**FM*****-ITRS-DE-P, R911336576 This documentation is provided with the hardware and describes the commissioning, the installation, the decommissioning and disposal of the hardware.

Tab. 1-12: XCL/MLC documentation overview - SafeLogic

Hardware
Rexroth IndraControl L45/L65/L85 Control DOK-CTRL-ICL45L65L85-PRxx-EN-P, R911332116 This documentation describes the IndraControl L45/L65/L85 controls.
Rexroth IndraControl L25 DOK-CTRL-IC*L25*****-PRxx-EN-P, R911328474 This documentation describes the IndraControl L25 controls.
Rexroth IndraControl Lxx 14VRS Function Modules DOK-CTRL-FM*LXX**V14-APRS-EN-P, R911341583 This documentation describes all function modules of the Lxx controls including engineering and diagnostics.
IndraControl XM21, XM22 Controls DOK-CTRL-IC*XM2*****-ITRS-EN-P, R911340667 This documentation describes the IndraControl XM21/XM22 controls.
IndraControl XFE 01.1 Extension Modules Profibus, RT-Ethernet, Sercos, CAN DOK-CTRL-XFE**EXTMOD-ITRS-EN-P, R911345570 This documentation describes all extension modules of the XM controls.

xx Corresponding edition
Tab. 1-13: XCL/MLC documentation overview - Hardware

About this documentation

Hydraulics
Rexroth IndraMotion MLC 14VRS Hydraulic Functions DOK-MLC***-TF*HMOT*V14-LIRS-EN-P, R911341907 This documentation describes the function blocks, functions and data types of the MH_TechHydrBase and MH_TechHydrMotion libraries. The documentation also includes the error reactions of the function blocks.
Rexroth IndraLogic XLC IndraMotion MLC 14VRS Sequential Programming DOK-XLCMLC-SEQPROG*V14-LIRS-EN-P, R911341909 This documentation describes the sequential programming options, also when using GAT, and the MH_HydrControlProg library.

Tab. 1-14: XCL/MLC documentation overview - Hydraulics

1.5 Required and supplementing documentation MTX

1.5.1 Selection/compilation

Documentation titles with type codes and part numbers

IndraMotion MTX 14VRS System Description DOK-MTX***-SYS*DES*V14-PRRS-EN-P, R911342640 This documentation describes the Rexroth IndraMotion MTX control. It includes the designs, technical data, interfaces as well as the configuration of the control components.
IndraMotion MTX 14VRS SafeLogic System Overview DOK-MTX***-SL**SYS*V14-PRRS-EN-P, R911341698 This documentation describes the use of the safety control SafeLogic in the IndraMotion MTX.

Tab. 1-15: MTX documentation overview - Selection/compilation

1.5.2 Configuration

Documentation titles with type codes and part numbers

IndraMotion MTX 14VRS Machine Parameters DOK-MTX***-MA*PAR**V14-RERS-EN-P, R911342630 This documentation describes design and modification of the Rexroth IndraMotion MTX parameters available. It also includes the functions of the NC configurator and its operation.
IndraMotion MTX 14VRS PLC Interface DOK-MTX***-PLC*INT*V14-PRRS-EN-P, R911342622 This documentation describes interface signals and program function blocks for the integrated PLC.
IndraMotion MTX 14VRS Functional Description - Basics DOK-MTX***-NC*F*BA*V14-RERS-EN-P, R911342626 This documentation describes the functions of the Rexroth IndraMotion MTX basic functions. The basic commissioning steps and the control functions are provided as description and handling instruction.
IndraMotion MTX 14VRS Functional Description - Extension DOK-MTX***-NC*F*EX*V14-RERS-EN-P, R911342628 This documentation describes the extended functions of the Rexroth IndraMotion MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.

IndraMotion MTX 14VRS Functional Description - Special Functions DOK-MTX***-NC*F*SP*V14-RERS-EN-P, R911342624 This documentation describes the special functions of the Rexroth IndraMotion MTX. The basic commissioning steps and the control functions are provided as description and handling instruction.
IndraMotion MTX Free Form Surface Milling DOK-MTX***-FREEFORM***-APRS-EN-P, R911341435 This documentation describes the free-form surface machining process with the MTX control. CNC programs generated by a CAD/CAM system are the basis for the entire machining process. An overview of the following topics is provided: Description of the MTX or CNC functions for free-form surface machining, NC parameterization, basic conditions when generating CNC programs with CAM software.
Rexroth IndraMotion MTX Converting MTX Projects DOK-MTX***-PROCONV****-PRRS-EN-P, R911342484 This documentation provides support during the conversion of IndraMotion MTX 1.x projects to MTX 2G. The conversion process is described and potential challenges during the conversion are identified.

Tab. 1-16: MTX documentation overview - Configuration

1.5.3 Commissioning

Documentation titles with type codes and part numbers

Rexroth IndraWorks 13VRS Simulation DOK-IWORKS-SIMU****V13-APRS-EN-P, R911339256 This documentation describes the functions of the simulation components View3D, virtual control panel, virtual control and its operation in IndraWorks.
IndraMotion MTX 14VRS Commissioning DOK-MTX***-STARTUP*V14-CORS-EN-P, R911342620 This documentation describes the commissioning of the IndraMotion MTX control. Apart from a complete overview, commissioning and configuration of the axes and the user interface as well as the PLC data are described.
IndraWorks 14VRS Basic Libraries IndraLogic 2G DOK-IL*2G*-BASLIB**V14-LIRS-EN-P, R911343920 This documentation describes the system-comprehensive PLC libraries.
IndraWorks 14VRS Field Buses DOK-IWORKS-FB*****V14-APRS-EN-P, R911341485 This documentation describes field bus and local periphery connections supported by the systems IndraLogic XLC, IndraMotion MLC and IndraMotion MTX. This documentation focuses on the configuration, parameterization, commissioning and diagnostics of the different periphery connections. The documentation is the basis for the online help.
IndraWorks 14VRS WinStudio 7.4 DOK-IWORKS-WINSTUD*V14-APRS-EN-P, R911341585 This "User Manual and Technical Reference Book" provides information for efficient working with the "Rexroth WinStudio"™ software. The document provides technical information and step-by-step instructions to create web-enabled HMI/SCADA programs.
IndraWorks 14VRS Software Installation DOK-IWORKS-SOFTINS*V14-CORS-EN-P, R911344286 This documentation describes the IndraWorks installation.
IndraWorks 14VRS Engineering DOK-IWORKS-ENGINEE*V14-APRS-EN-P, R911343566 This documentation describes the use of IndraWorks in which the Rexroth Engineering tools are integrated. It includes instructions on how to work with IndraWorks and how to operate the oscilloscope function.

About this documentation

Rexroth IndraWorks 12VRS FDT Container

DOK-IWORKS-FDT*CON*V12-APRS-EN-P, R911334398

This documentation describes the IndraWorks FDT Container functionality. It includes the activation of the functionality in the project and working with DTMs.

IndraWorks 14VRS PLC Programming System IndraLogic 2G

DOK-IWORKS-IL2GPRO*V14-APRS-EN-P, R911343571

This documentation describes the PLC programming tool IndraLogic 2G and its use. It includes the basic use, first steps, visualization, menu items and editors.

IndraWorks 14VRS HMI

DOK-IWORKS-HMI*****V14-APRS-EN-P, R911343569

This documentation describes the functions, configuration and operation of the user interfaces IndraWorks HMI Engineering and IndraWorks HMI Operation.

Tab. 1-17: MTX documentation overview - Commissioning

1.5.4 Operation

Documentation titles with type codes and part numbers

Rexroth IndraMotion MTX 09VRS Shop Floor Programming Turning and Milling

DOK-MTX***-SF*PROG*V09-AWxx-EN-P, R911324377

This documentation describes the operation and programming of the graphic NC programming of turning and milling as well as the workpiece simulation.

Rexroth IndraMotion MTX 12VRS Block Pre-Run

DOK-MTX***-BLK*RUN*V12-APRS-EN-P, R911334379

This documentation explains to the machine manufacturer how to setup the "Block pre-run" function at the machine for the end user.

IndraMotion MTX 14VRS Programming Manual

DOK-MTX***-NC**PRO*V14-RERS-EN-P, R911342634

This documentation describes the standard programming of the Rexroth IndraMotion MTX control. Apart from the basics of NC programming, the use of NC functions according to DIN 66025 as well as the NC functions with high-level language syntax and CPL functions are described.

IndraMotion MTX 14VRS Standard NC Operation

DOK-MTX***-NC*OP***V14-APRS-EN-P, R911342632

This documentation describes the operation of the standard user interface of the NC control of the Rexroth IndraMotion MTX. It includes the operation of the interface, the NC program development as well as the tool management.

IndraMotion MTX 14VRS Standard NC Cycles

DOK-MTX***-NC*CYC**V14-PRRS-EN-P, R911342638

This documentation describes the application of the standard cycles of the different technologies for Rexroth IndraMotion MTX control.

IndraMotion MTX 14VRS Multitouch

DOK-MTX***-MULTI***V14-APRS-EN-P, R911384735

The Multitouch user interface of the NC control Rexroth IndraMotion MTX is described in this documentation.

IndraMotion MTX 14VRS NC Simulation

DOK-MTX***-NC*SIM**V14-APRS-EN-P, R911379629

This documentation describes the NC Simulation for Rexroth IndraMotion MTX control.

xx

Corresponding edition

Tab. 1-18: MTX documentation overview - Operation

1.5.5 OEM Engineering

Documentation titles with type codes and part numbers

Rexroth IndraMotion MTX 13VRS Automation Interface DOK-MTX***-AUT*INT*V13-APRS-EN-P, R911337274 This documentation describes the script-based access to IndraWorks project data via the Automation Interface. Different objects including code examples are described. The Automation Builder is also described in this manual.
Rexroth IndraMotion MTX 14VRS OPC Communication DOK-MTX***-OPC*COM*V14-PRRS-EN-P, R911342636 This documentation describes the syntax and the structure of the items for the communication with Bosch Rexroth devices.
Rexroth IndraMotion MTX 14VRS OPC UA Communication DOK-MTX***-OPC*UA**V14-APRS-EN-P, R911342513 This documentation describes the OPC UA communication of the IndraMotion MTX.

Tab. 1-19: MTX documentation overview - OEM Engineering

1.5.6 AddOns

Documentation titles with type codes and part numbers

Rexroth IndraMotion MTX 11VRS Action Recorder DOK-MTX***-ACR*****V11-APxx-EN-P, R911329943 This documentation describes the MTX action recorder. It includes the installation and commissioning as well as interface signals, application and operation.
Rexroth IndraMotion MTX 14VRS efficiency workbench MTX cta, MTX ega DOK-MTX***-EWB*****V14-APRS-EN-P, R911343916 This documentation describes the mode of operation and the area of application of the analysis tool IndraMotion MTX cta and IndraMotion MTX ega.
Rexroth IndraMotion MTX Remote Condition Monitoring DOK-MTX***-RCM*****V01-APRS-EN-P, R911334383 This documentation describes the operation of the Remote Condition Monitoring System.
Rexroth IndraMotion MTX visIREC User Documentation DOK-MTX***-VISIREC*V01-APRS-EN-P, R911344242 This documentation describes the analysis tool visIREC. The visIREC is used to optimize the free-form surface machining process. 2D or 4D display of path-related data. 2D or 4D display of coordinate-related data. Analyzing critical areas (path and orientation deviation). Comparing the programmed and interpolated NC blocks.

xx Corresponding edition

Tab. 1-20: MTX documentation overview - AddOns

1.6 Names and abbreviations

Name	Explanation
IndraMotion MLC	Compact Motion Logic system with Motion, Robot and Logic Control functionality
IndraLogic XLC	Compact PLC system with integrated Motion functionality
IndraMotion MTX	Compact CNC Motion system with CNC Motion and Logic Control functionality

About this documentation

AxisInterface	The AxisInterface bundles and extends PLCopen motion function blocks and provides a user-friendly interface for the drive functionality (only for XLC/MLC)
Function block	Provides application-specific functionalities
IndraWorks Engineering Framework	Project planning and commissioning tool of Bosch Rexroth
IndraDrive	Drive control device
Wizard	Wizard guiding through several dialogs for an ergonomic data input
CrossComm	Cross communication
Profibus	Profibus (Process Field Bus) is a standard for the field bus communication in automation engineering
PROFINET	Profinet (Process Field Network) is an open industrial Ethernet standard of Profibus & Profinet International (PI) for the automation. PROFINET uses TCP/IP and IT standards. It is real-time Ethernet-compatible and allows the integration of field bus systems
EtherNet/IP	EtherNet/IP (EtherNet Industrial Protocol, often also called EIP) is a field bus based on the Ethernet. This field bus is mainly used in automation technology
TCP/IP	Transmission Control Protocol/Internet Protocol (TCP/IP) is a family of network protocols
IEC	International Electrotechnical Commission
Sercos	Sercos (SErial Realtime COmmunication System) interface is a world-wide standardized interface for the communication between controls and drives
SRAM	Static Random Access Memory (static RAM)

Tab. 1-21: Names and abbreviations used

1.7 Customer feedback

Customer requests, comments or suggestions for improvement are of great importance to us. Please email your feedback on the documentations to Feedback.Documentation@boschrexroth.de. Directly insert comments in the electronic PDF document and send the PDF file to Bosch Rexroth.

2 Introduction

Nowadays in particular, an economical use of resources is an important issue in private households as well as in an industrial environment. This topic has a high priority, not only from an ecological point of view but also in terms of cost effectiveness.

This aspect also has to be taken into consideration when operating machines. Bosch Rexroth developed the 4EE system in order to detect and use significant energy saving potentials in industrial systems. 4EE is structured into four strategies:

1. **Energy System Design** (systematic overall assessment, project planning, simulation and consultation)
2. **Efficient Components** (products and systems with optimized efficiency)
3. **Energy Recovery** (recovery and storage of excess energy)
4. **Energy on Demand** (demand-driven use of energy, standby mode)

By taking into consideration the four mentioned points, the energy efficiency can be increased sustainably during all phases of the machine life cycle. Operating costs are reduced and the productivity is increased. The 4EE strategy helps to secure advantages in competition, conserving resources and to reduce emissions. The energy efficiency is of vital importance in contributing to a sustainable and responsible use of environmental resources.

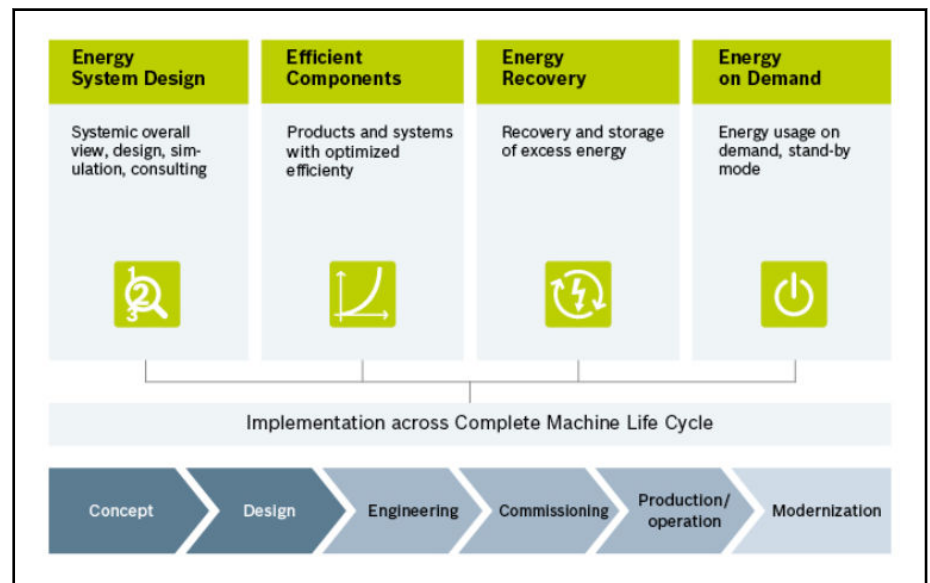


Fig. 2-1: 4EE system

3 EnergyManagement

3.1 Overview

The EnergyManagement function blocks (Control and Monitoring) have been developed to implement a demand driven energy use (Energy on Demand). They can be used with the systems MLC, XLC and MTX and configured and linked at the specific project as per the present conditions. By targeted disabling of consumers during downtimes (e.g. break times or operating states in which only certain consumers are used) facilitate a reduced energy consumption.

The 4EE EnergyManagement consists of two parts:

- EnergyMonitoring to determine energy consumption and power peaks
- EnergyControl to reduce the energy consumption by disabling consumers that are not required

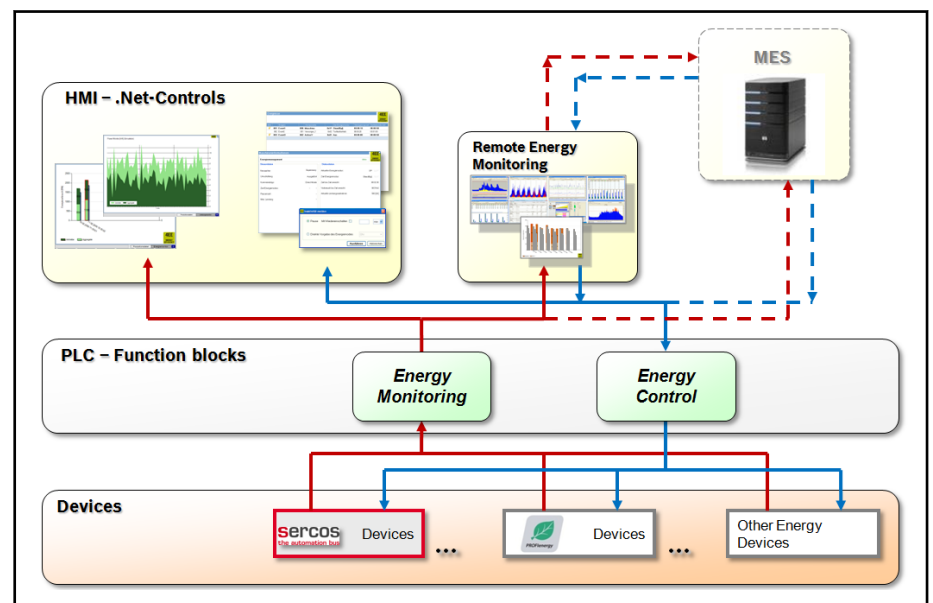


Fig. 3-1: EnergyManagement overview

The 4EE EnergyManagement provides a standardized, quick and reliable integration and commissioning of energy-saving functionalities. The 4EE EnergyManagement has the following advantages for the machine operator:

- Transparency of the energy consumption of the machine and the components
- Minimum energy consumption of its machines and systems in all situations
- Optional coupling option with manufacturing execution systems (MES)

The function blocks required for 4EE EnergyManagement are included in the RIL_4EEEnergyManagement (EnergyControl) and MT_4EnergyEfficiency (EnergyMonitoring) libraries.

For a detailed description of the function blocks, refer to [chapter 9 "RIL_4EEEnergyManagement.library"](#) on page 111.

EnergyManagement

EnergyControl area

Function block name	Description
IL_4EEEnergyControlType01, page 212	The central function block of the EnergyManagement ("control center") manages specifications of different command function blocks and passes them on to the consumers
IL_4EEBasicCommandType01, page 221	Specification of energy modes and break times
IL_4EEEventCommandType01, page 228	Specification of fixed energy modes and configurable delay times
IL_4EEEventCommandType02, page 235	Specification of fixed energy modes for compiled EnergyManagement programs. This function block can be registered in Device and EnergyMonitor function blocks at program runtime
IL_4EEBasicDeviceType01, page 242	Specification of the energy mode for individual consumers (i.e., establishing a connection between consumers and EnergyManagement)
IL_4EEAdvancedDeviceType01, page 248	Specification of the energy mode for individual consumers (i.e., establishing a connection between consumers and EnergyManagement). The function block supports the configuration of energy modes and thus individual calculation of the energy mode for break times
IL_4EEEnergyFunctionType01, page 255	It is used to visualize analog values of existing energy saving function (e.g. performance value of a fan in percent)
IL_4EEHMIDataCollectorType01, page 257	Bundles data for HMI
IL_4EEScadaDataCollectorType01, page 259	Bundles data for SCADA and ME systems

Tab. 3-1: Function blocks of the RIL_4EEEnergyManagement library in the EnergyControl area

EnergyMonitoring area

Function block name	Description
IL_4EEEnergyMonitorType01, page 113	Collects energy and performance data of consumers and provides the data to the Collector function block
IL_4EEMonBasicDeviceType01, page 115	Recording performance or energy data for general consumers
IL_4EEMonIndraDriveDeviceType01, page 120	Reading out the energy counter of IndraDrive drives
IL_4EEMonSercosEnergyDeviceType01, page 124	Reading out the energy counter of devices supporting the Sercos Energy Monitoring functionality
IL_4EEMonProfiEnergyDeviceType01, page 128	Reading out the energy counter of devices supporting the PROFIenergy Monitoring functionality
IL_4EEMonPowerSupplyDeviceType01, page 133	Reading out the energy counter of devices supporting the Sercos supplier profile (S-0-1702.0.14)
IL_4EEMonTerminalDeviceType01, page 137	Reading out the energy counter of the performance measuring terminals of Phoenix Contact
IL_4EEMonTerminalDeviceType02, page 142	Reading out the energy counter of the performance measuring terminals of Phoenix Contact with additional output of different measuring objects

IL_4EEMonCompAirDeviceType01, page 148	Function block to generically record energy and performance data for consumers of compressed air
IL_4EEMonFluidDeviceType01, page 153	Function block to generically record energy and performance data for consumers of fluids and gases where energies are transmitted via temperature differences (e.g. cooling water, heating gas)
IL_4EEMonGasDeviceType01, page 158	Function block to generically record energy and performance data for consumers of combustible gases
IL_4EEMonGasDeviceType02, page 163	Function block to generically record energy and performance data for consumers of combustible gases
IL_4EEMonPneumaticDeviceType01, page 168	Function block to generically record energy and performance data of pneumatic cylinders
IL_4EEMonS0DeviceType01, page 174	Function block to analyze energy consumption values transmitted via a pulse interface acc. to EN 62053-31 (S0 interface)
IL_4EELoadProfileEstimation, page 178	Function block to create a load profile prognosis on incoming energy consumption values
IL_4EEMonHMIDataCollectorType01, page 183	Executing energy measurements and data preparation for HMI (.NET controls)
IL_4EEMonHMIDataCollectorType02, page 186	Executing energy measurements and data preparation for web visualization (without language switching)
IL_4EEMonHMIDataCollectorType03, page 188	Executing energy measurements and data preparation for web visualization (with language switching)
IL_4EEMonScadaDataCollectorType01, page 190	Executing energy measurements and data preparation for SCADA and ME systems

Tab. 3-2: *Function blocks of the RIL_4EEEnergyManagement library in the EnergyMonitoring area*

The function blocks can be configured and connected in series according to the initial situation at the machine.

3.2 EnergyMonitoring

3.2.1 Motivation

EnergyMonitoring allows the recording of the energy consumption and the drive as well as ancillary units (heating, fan, hydraulic aggregates, etc.) to monitor or determine potential savings and load peaks. Together with interface applications "Energy and performance monitor" on the IndraWorks Operation Desktop, the results can be represented graphically.

3.2.2 Next steps

The following points have to be taken into consideration when using EnergyMonitoring.

1. Which consumers are available?
 - IndraDrive drives
 - Asynchronous motors
 - Heatings
 - ...
2. Which energy data is available?
 - Energy counter in device

EnergyManagement

- Nominal power according to type plate
- Switch-on times
- Controlled variables in percent

3.2.3 Functionality

The EnergyMonitoring functionality contains function blocks for managing and calculating energy modes for individual consumers in a machine and their visualization.

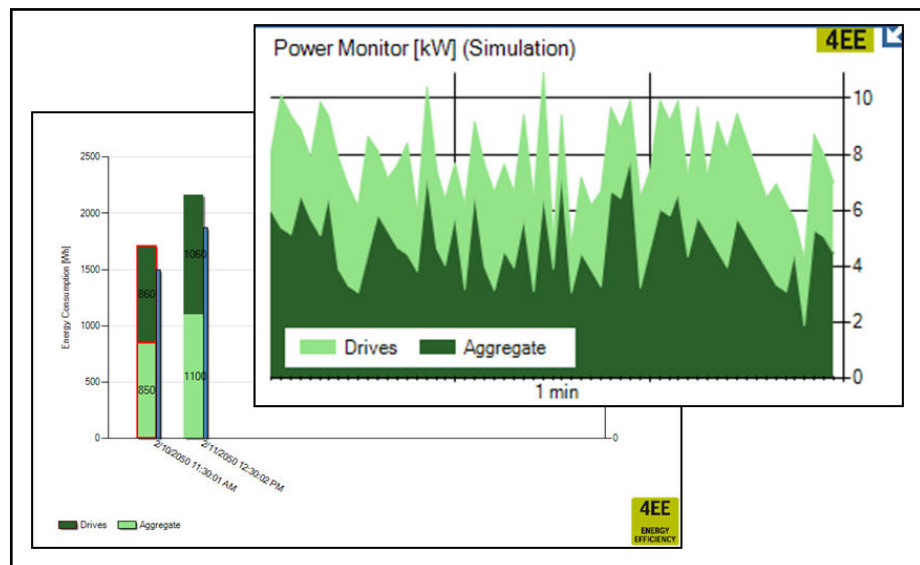


Fig. 3-2: EnergyMonitoring

3.2.4 Architecture

The individual EnergyMonitoring function blocks are connected via the declaration and communicate via mechanisms of the object-oriented extension. The function block interconnection is explained in detail in [chapter 6.2 "Declarations and nesting of the function blocks" on page 49](#). The HMICollector function block provides the energy and performance data as well as the required configuration to the controls via global structures.

EnergyManagement

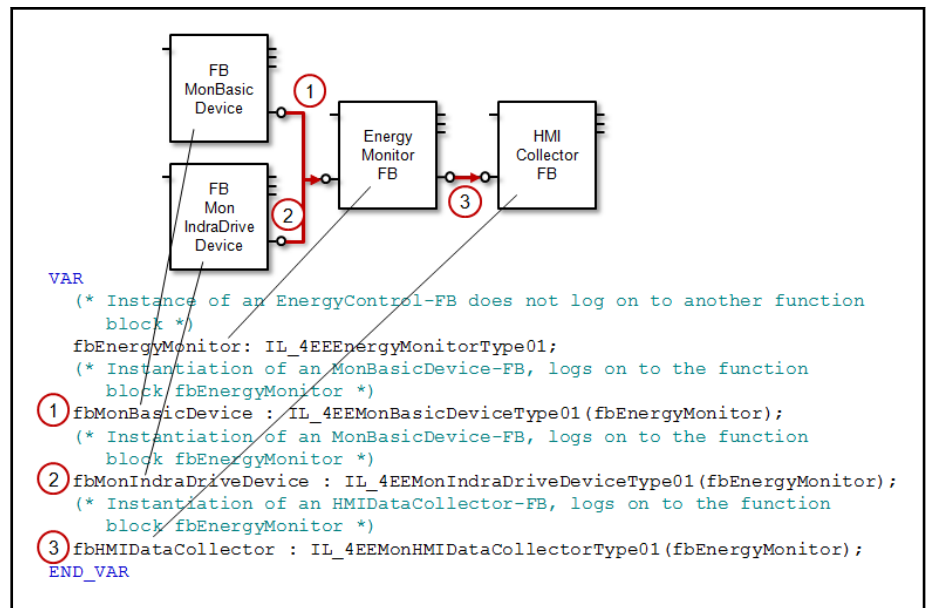


Fig. 3-3: EnergyMonitoring architecture

3.2.5 Functioning

The MonitoringDevice function blocks read or calculate the energy data of the consumers. The EnergyMonitor function block collects the energy data and calculates the performance data of the consumers. The HMICollector function block prepares the data for the HMI. Furthermore, energy measurements can be started and stopped using the function block.

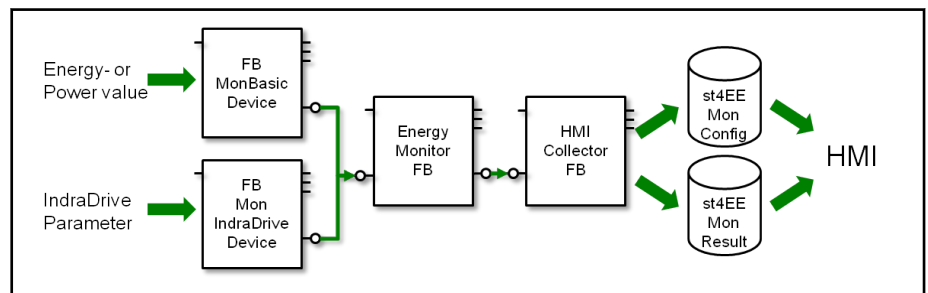


Fig. 3-4: EnergyMonitoring functioning

3.3 EnergyControl

3.3.1 Motivation

The EnergyControl facilitates a reduction of the energy consumption by targeted switching of consumers in energetically more favorable conditions in case of scheduled and unscheduled break times.

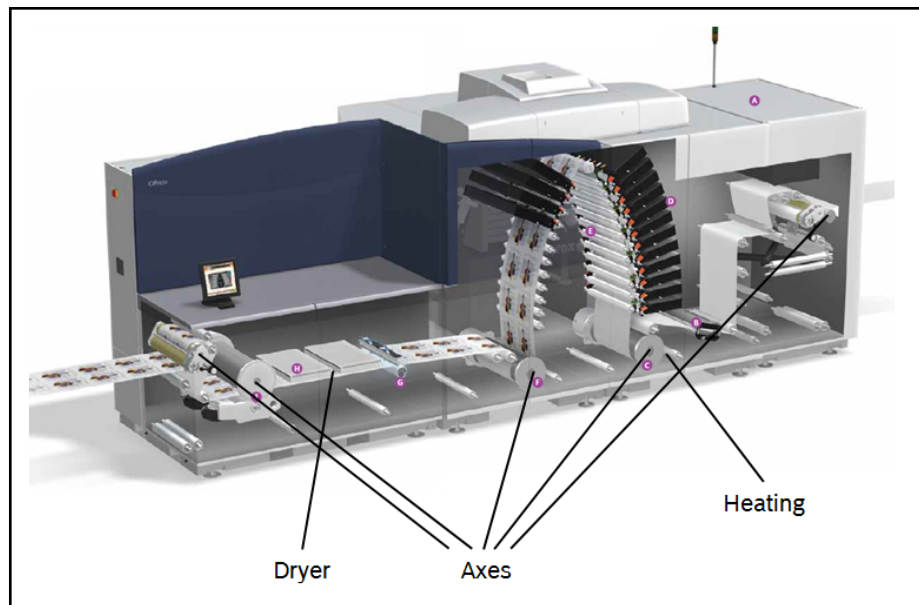


Fig. 3-5: Printing press, hardware view

3.3.2 Next steps

The following points have to be taken into consideration when using Energy-Control:

1. Are break times set?
 - Weekends, breakfast and lunch breaks, cyclic breaks or unscheduled breaks
2. Which parts of the system and consumers can be switched off?
 - Does media have to be heated or cooled during break times?
 - Do hanging axes exist?
 - Do safety-relevant aspects have to be taken into consideration?
 - Are components not permanently needed during operation?
3. Which components are used?
 - Sercos Energy, PROFIenergy or consumer without energy profile.

3.3.3 Sercos Energy

With Sercos Energy, controls can set connected I/O components to energy-saving status - up to the complete switch-off - and thus, reduce the energy consumption considerably. Sercos Energy takes energy-saving status for scheduled break times such as breakfast and lunch breaks as well as plant holidays into consideration. At predefined times, the Sercos Energy components are deliberately set to idle state to save energy. Sercos Energy continues to provide mechanisms for unscheduled downtimes, created, for example by a shortage of parts. The following user scenarios are defined for the master and machine level for Sercos Energy:

- **Short break times**
In case of short break times with a duration of up to one hour, the machine is switched to the best energetic state.
- **Long break times**
Long dead times are break times where the duration is already known, ranging from several hours up to several days or even weeks. As for

short breaks, the machine components are set to the ideal energetic state.

- **Unscheduled break times**

Unscheduled break times are dead times of unknown duration. These dead times are often created by a shortage of parts or operator intervention.

- **Restricted machine operation**

Due to the restricted machine operation in Sercos Energy, machine components currently not required in the process are switched off.

- **Partial load operation**

If a certain maximum performance must not be exceeded during operation, a limit can be specified in the machine.

Sercos Energy is the basis of the EnergyManagement. The following command types are defined for Sercos Energy:

- Direct energy mode specification
- Break time specification
- Specifying a break time with automatic reactivation

3.3.4 Functionality

The EnergyControl functionality contains function blocks for managing and calculating energy modes for individual consumers in a machine and their visualization.

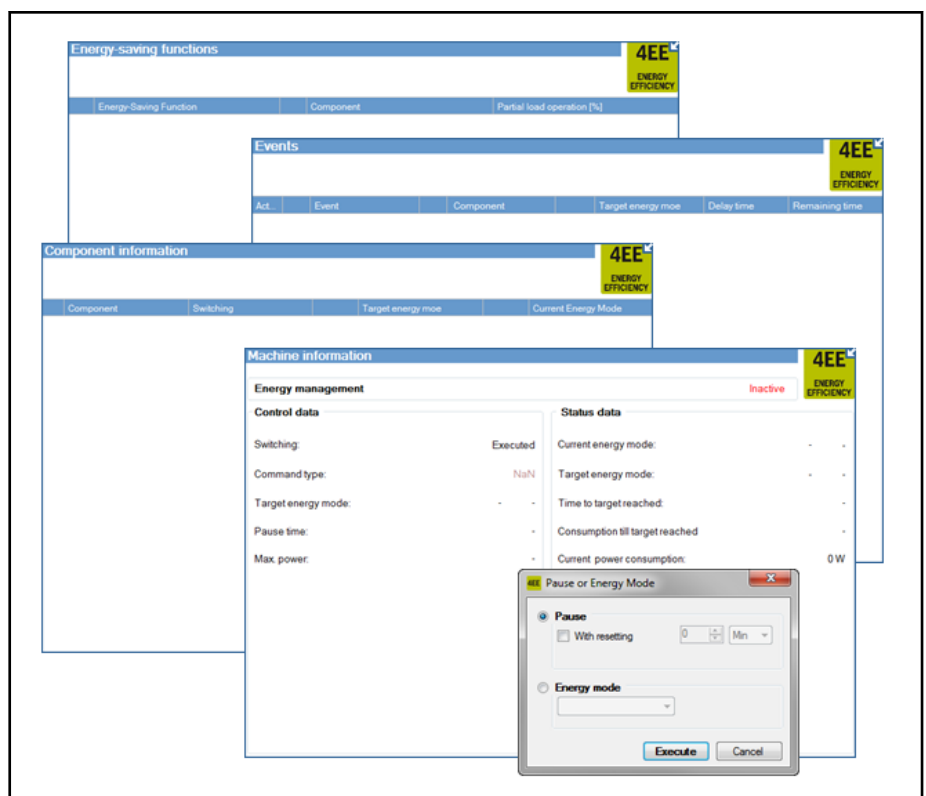


Fig. 3-6: EnergyControl in the user interface

3.3.5 Architecture

The individual EnergyControl function blocks are connected via the declaration and communicate via mechanisms of the object-oriented extension. The

EnergyManagement

function block interconnection is explained in detail in [chapter 7.2 "Declarations and nesting of the function blocks"](#) on page 55.

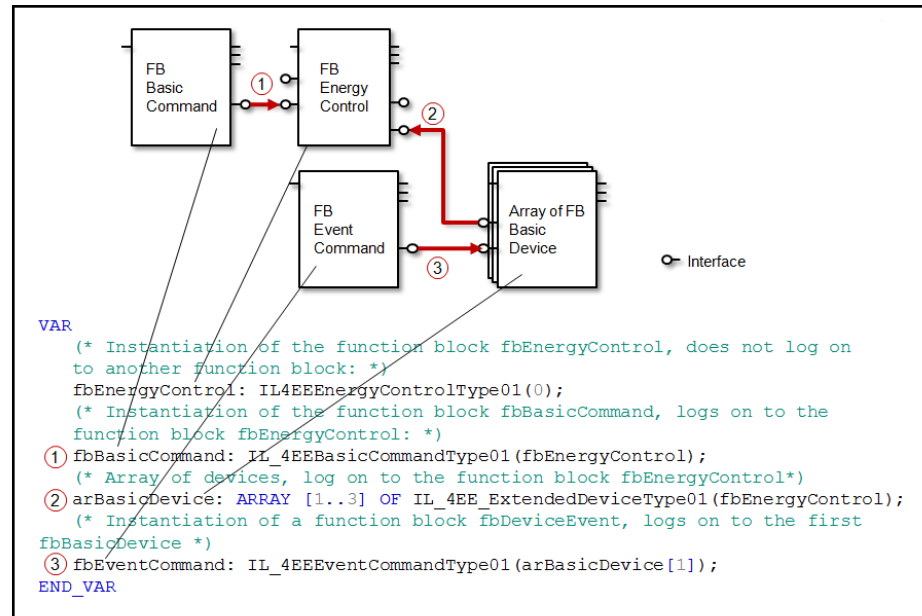


Fig. 3-7: EnergyControl architecture

3.3.6 Functioning

An EnergyControl function block for managing and calculating the energy modes is the central component of EnergyControl. The Device function blocks are registered via interfaces in the EnergyControl function blocks. The Device function blocks contain the configuration of the individual consumers and manage their specific energy modes. For EnergyManagement specifications that should apply to all devices, the command blocks are registered in the EnergyControl function block.

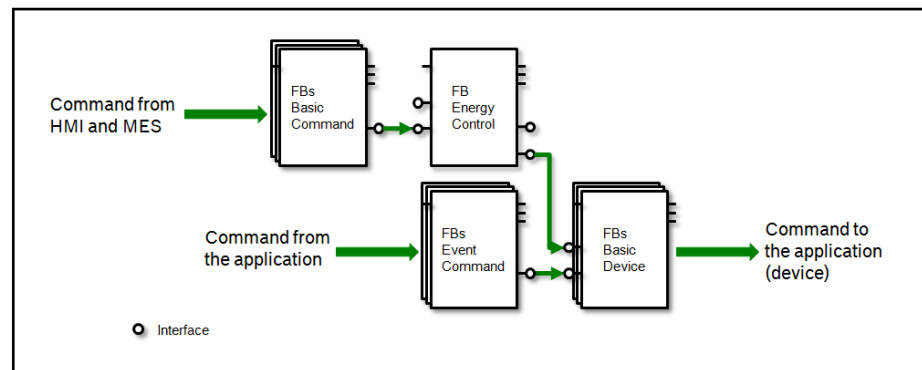


Fig. 3-8: Structure of EnergyControl function blocks

3.3.7 Modularization

Modules and groups can be created by cascading EnergyControl function blocks.

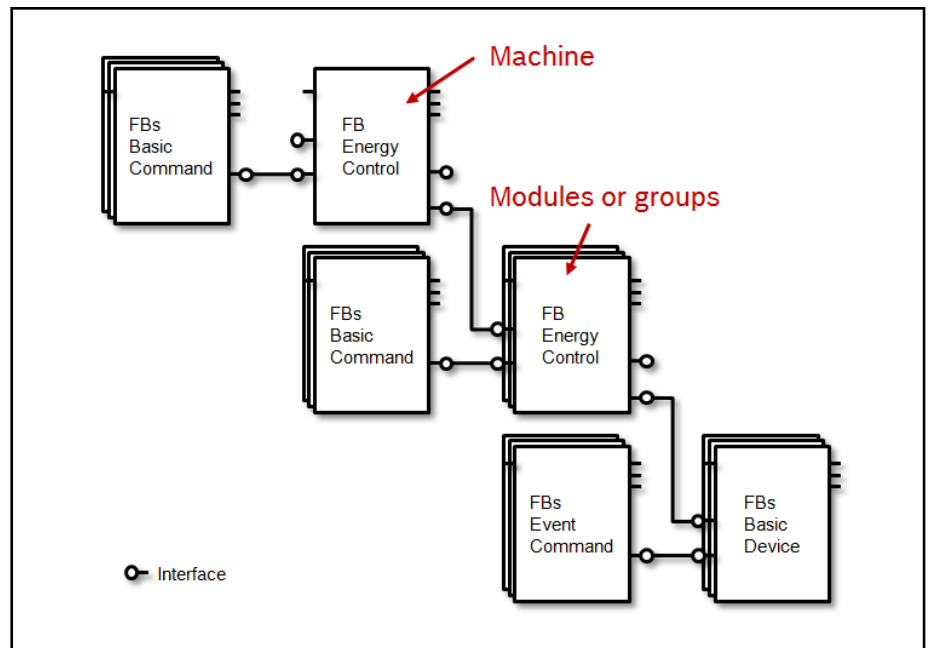


Fig. 3-9: EnergyControl modularization

3.3.8 Coupling

The mGetControl method queries the current target energy mode. The current status is returned using the mSetStatus method. Due to a number of different consumers as well as in particular the applications, generic switching is not possible. Thus, the PLC user has to forward the results of the Device function block for each consumer to the device, taken the given conditions into consideration, and to reports back its status.

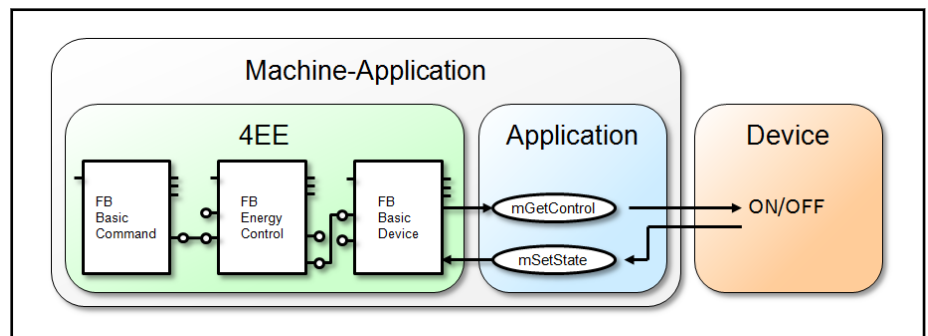


Fig. 3-10: EnergyControl coupling

4 Commissioning example: Handling and press

4.1 Overview

The machine produces an exemplary key-shaped USB flash drive. Two different production steps are executed at two stations:

1. **Handling:** The USB flash drive is equipped with a small plastic plate.
2. **Pressing:** The USB flash drive equipped with a small plastic plate runs through a press and the plastic plate is pressed on the USB flash drive.

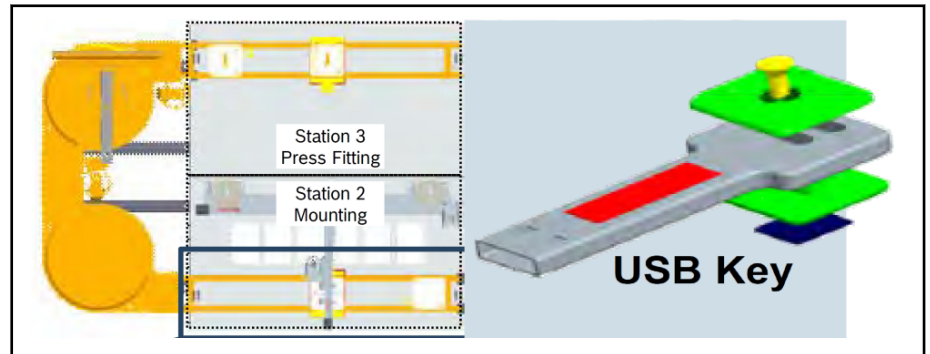


Fig. 4-1: Stations and workpiece

4.2 Module: Handling

During handling, the workpiece is transported to the processing location on a conveyor belt. Different small plastic plates are available at the processing location. The USB flash drive is equipped by means of three linear axes (X-, Y- and Z-axis). The small plastic plates are positioned on the upper and lower side of the workpiece. Each linear axis constitutes a consumer. The conveyor belt subsequently transports the processed workpiece. A lamp illuminates the station. Thus, five different consumers are part of the process:

1. **Conveyor belt**
2. **Linear motor X-axis**
3. **Linear motor Y-axis**
4. **Linear motor Z-axis**
5. **Lighting**

Commissioning example: Handling and press

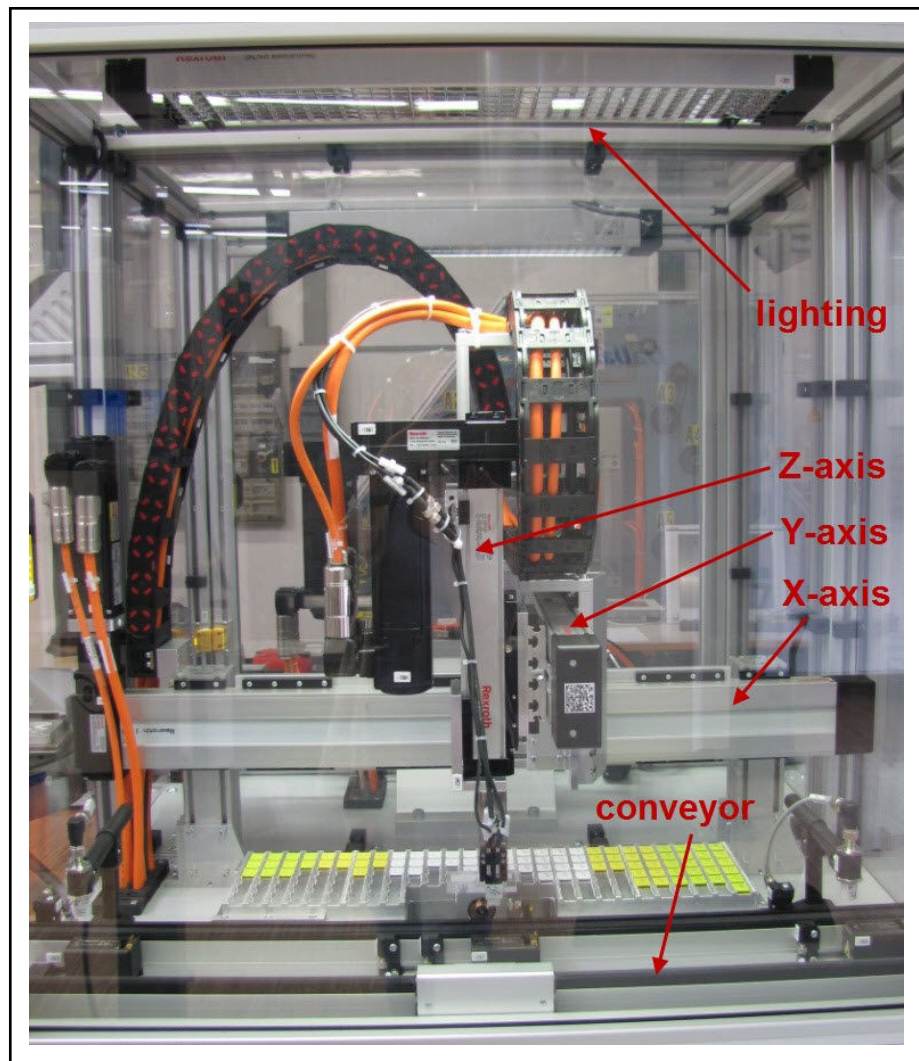


Fig. 4-2: Module: Handling

4.3 Module: Press

The small plastic plates positioned on the workpiece are now pressed on. The workpiece is transported to the press on the conveyor belt. The press is lowered and presses the small plastic plate onto the USB flash drive. The conveyor belt subsequently transports the workpiece. This station is also illuminated. Thus, there are three different consumers:

1. Conveyor belt
2. Press motor
3. Lighting

Commissioning example: Handling and press

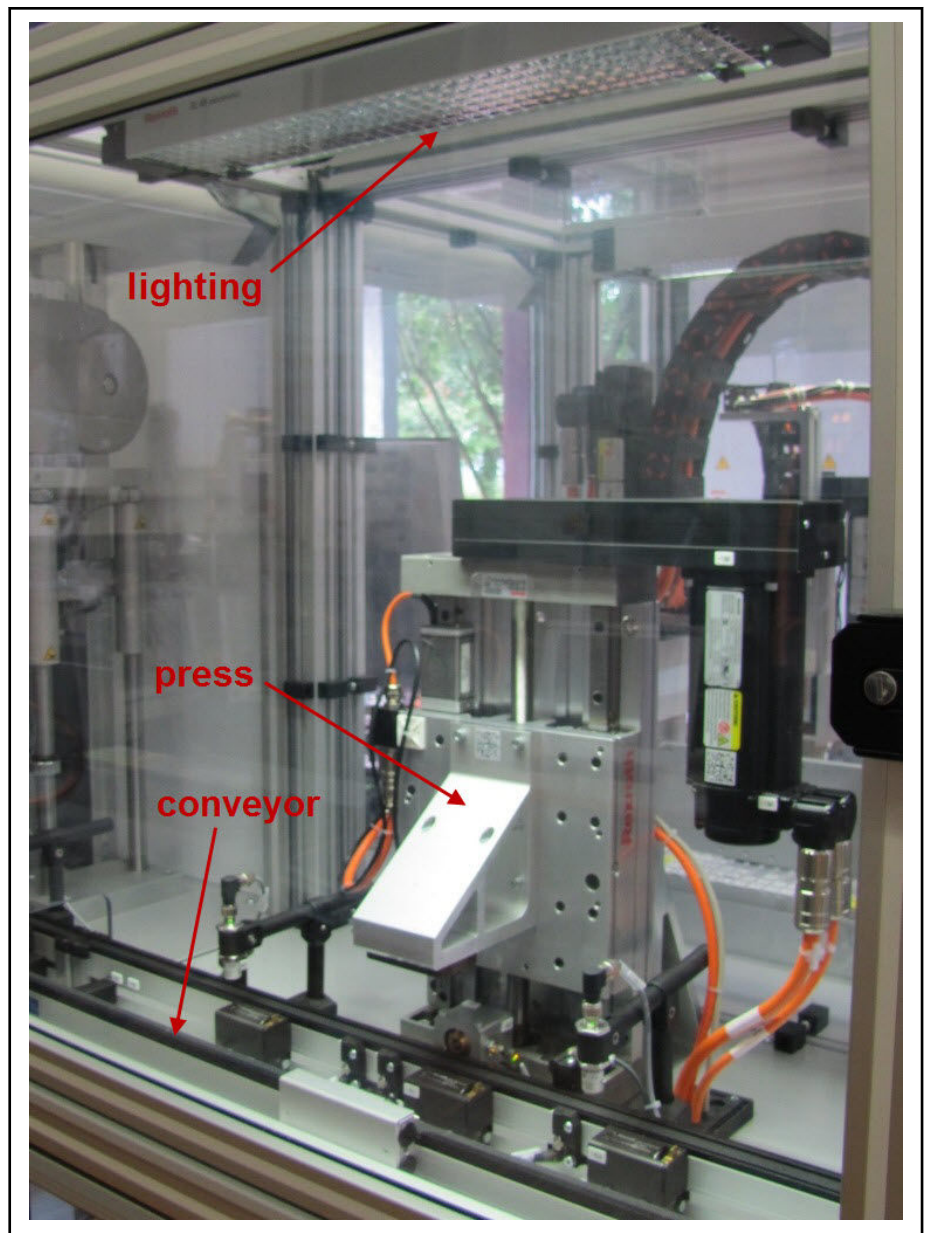


Fig. 4-3: Module: Press

5 Configuring the EnergyManagement function blocks

5.1 Overview

A solution for EnergyMonitoring and EnergyControl is described below for the machine described in [chapter 4 "Commissioning example: Handling and press"](#) on page 31.

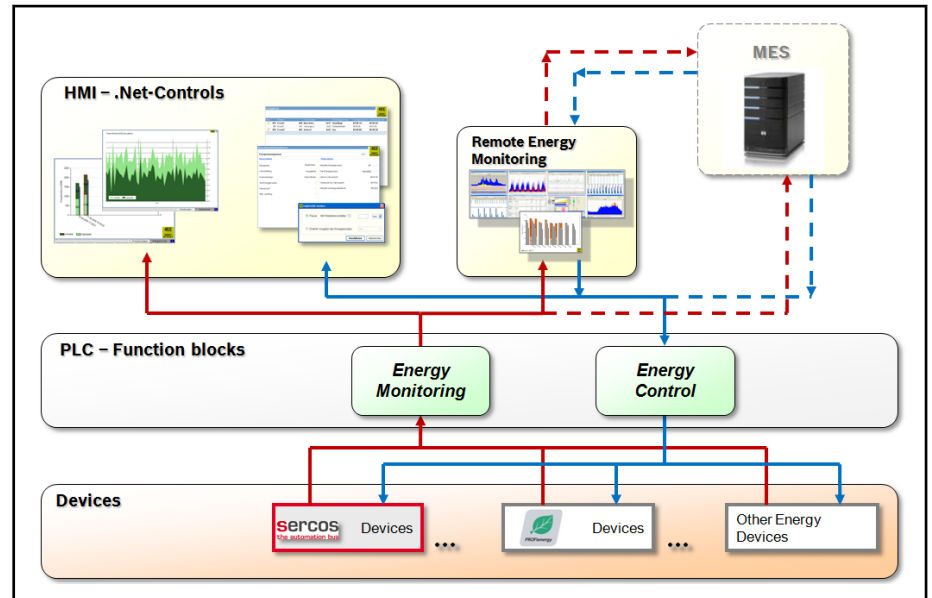


Fig. 5-1: Overview: EnergyManagement

5.2 EnergyMonitoring

5.2.1 Overview

Handling and press are considered a single machine in the representation with function blocks. Monitoring currently does not differentiate between the modules. Which function blocks are actually needed and how these should be connected now has to be considered.

5.2.2 Handling and press

An EnergyMonitor block for which the blocks are registered for each consumer is needed for the machine. The axes of each consumer must be differentiated from one another, as well as each consumer from other consumers. The function block IL_4EE_MonIndraDriveDevice is used for the axes. Of these, four instances are required for the X-, Y- and Z-axes, as well as for the press. The function block IL_4EEMonBasicDeviceType01 is used for recording the lighting and the conveyor belts. Both belts and both lights also require four instances.

Another instance of the function block IL_4EEMonHMIDataCollectorType01 is also needed for the HMI display. The energy measurements can also be started and stopped with this function block. Along with the MonitoringDevice blocks, this instance is also registered with the EnergyMonitor block.

Configuring the EnergyManagement function blocks

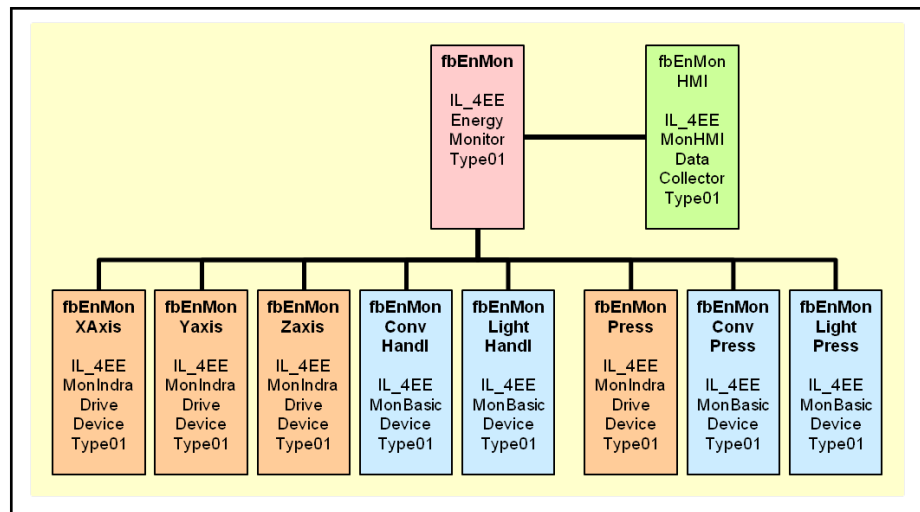


Fig. 5-2: Connecting function blocks for EnergyMonitoring

5.3 EnergyControl

5.3.1 Setting the energy modes for machine and stations

A control structure that includes all consumers and enables practical specifications regarding energy modes to be set should be drafted from EnergyControl blocks below for operating both stations.

Various situations are conceivable in which consumers should be switched on and off:

- **Normal operation:** All consumers are in operation. The workpieces conveyed to the stations have small plastic plates. These are pressed on in the next work step.
- **Dead time:** The machine should not be producing during employee break times, time between shifts, and on weekends. During this dead time, no consumer should be in operation, and all consumers incl. lighting can be switched off.
- **Single station operation (Handling or Press):** It can be practical for various reasons to only operate one of the two stations (Handling or Press), e.g., if the workpieces are not pressed, but removed at the Handling station immediately after assembly (e.g., when using special, self-adhering plastic plates). It would then be practical to switch off the entire Press station. It is also possible that assembled workpieces only need to be pressed. In this instance, the Handling station should be able to be completely switched off.
- **Machine standstill with light on:** This can be practical if no workpieces are currently awaiting processing. In order to guarantee that the machine automatically switches off in this situation, the conveyor belt can be equipped with a sensor. It registers that no workpieces are currently approaching. In order to be able to closely inspect the stations, e.g., during maintenance, good lighting is beneficial. This is why the two lamps remain on, while all other consumers are not needed and can be switched off.
- **Transporting the workpiece without assembly and pressing:** If the USB flash drive should not have a plastic plate, the three axes and the press motor are not needed and can be switched off. The conveyor belt remains needed for transport, and the lighting should also remain on in order to allow the workpiece to be inspected. The same operating mode is

Configuring the EnergyManagement function blocks

practical if there is a gap between workpieces and new workpieces are expected on the belt for processing.

5.3.2 Safety

The Handling and Press drive axes can only be switched off in this sample application, since they have fixing brakes. Otherwise this could result in damage.

5.3.3 Specifications and commands

In this example, three energy modes are defined for the machine. The following figure shows these three operating situations and how the consumers are connected.

Energy modes								
Module No	Module Name	Device Nr.	Device Name	Modes				Description
				0	1	2	3	
				Operational	Ready for Power	Standby		
1	Handling	1	Conveyor Handl	on	on	off		Conveyor of the handling module
		2	X-axis	on	off	off		Linear motor for the X-axis
		3	Y-axis	on	off	off		Linear motor for the Y-axis
		4	Z-axis	on	off	off		Linear motor for the Z-axis
		5	Lighting Handl	on	on	off		Lighting of the handling module
2	Press	6	Conveyor Press	on	on	off		Conveyor of the press module
		7	Press	on	off	off		Press motor
		8	Lighting Press	on	on	off		Lighting of the press module

Fig. 5-3: Various operating situations and consumer circuitry

The following energy modes can be configured for the linear motors for the X-, Y- and Z-axes, as well as for the Press:

- **Standby:** No control enable (AB)
- **Ready for Power:** No control enable (AB)
- **Operational:** Drives in control mode (AF)

In actuality, the conveyor belt and lighting for handling and press only have the states "On" and "Off". Therefore, only two different energy modes are possible:

- **Standby:** Consumer is off
- **Operational:** Operating mode

5.3.4 Handling

First, the Handling station should be represented by function blocks. As described above, it should be possible to turn the entire Handling module on and off, or to specify dead times for the module. This is possible with a Basic-Command block. The EnergyManagement specifications are transmitted to the consumers using Device blocks. One Device block is needed for each consumer. This means the following blocks are always needed:

Configuring the EnergyManagement function blocks

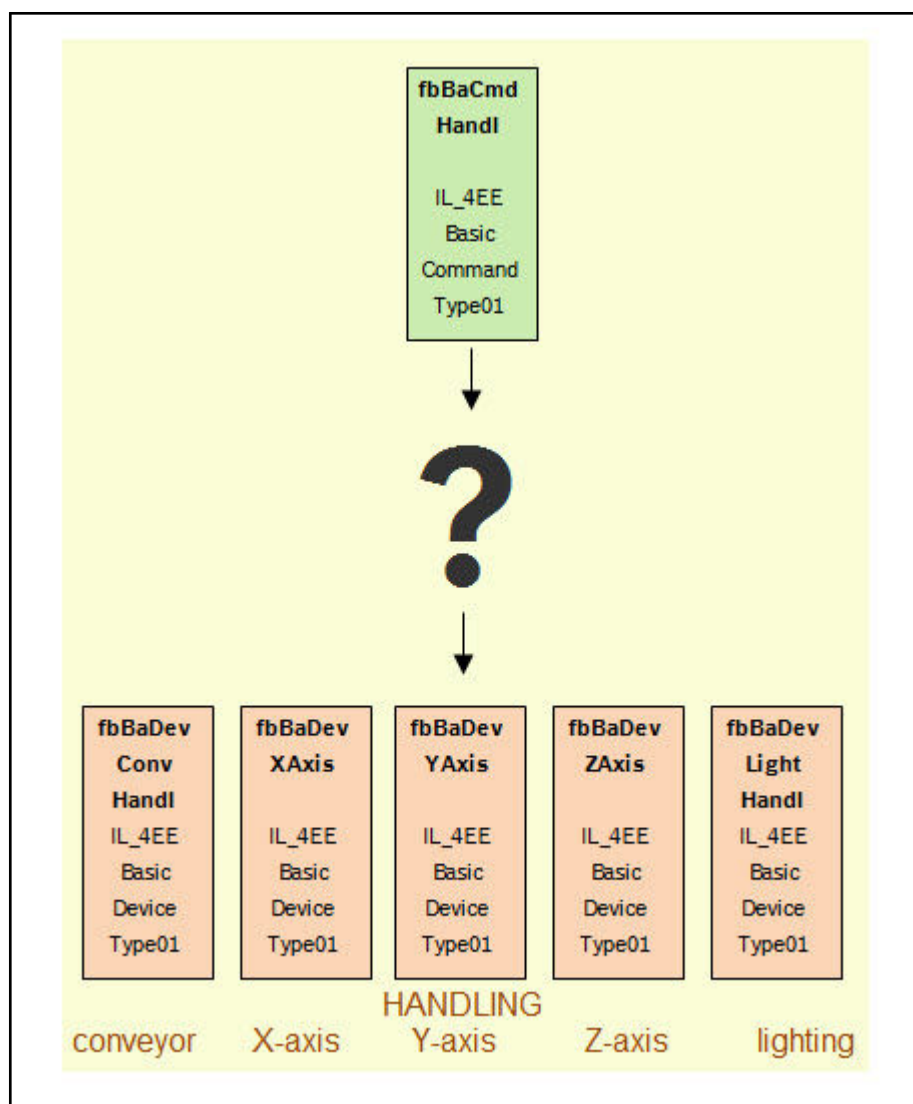


Fig. 5-4: BasicCommand and BasicDevice blocks for Handling module

The BasicCommand block now has to be linked to the BasicDevice blocks. An EnergyControl block is used to do this. It transmits the specifications of the BasicCommand block to the Device blocks:

Configuring the EnergyManagement function blocks

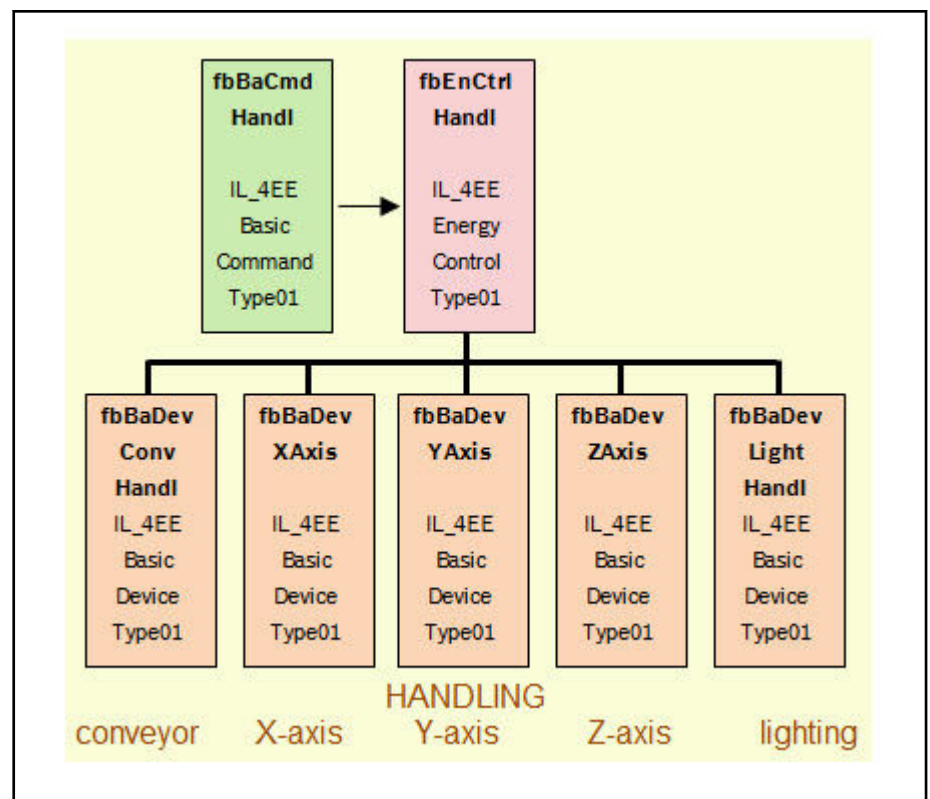


Fig. 5-5: Handling module - Basic structure

There is now an arrangement of function blocks that make it possible to turn all Handling consumers on and off at once, or to specify dead times for the entire module. Next, the sensor showing the incoming workpieces should be linked to EnergyManagement. If no workpieces are awaiting processing, the three handling axes and the conveyor belt should be turned off. EventCommand blocks are suited for specifying precise energy modes for individual consumers, as is required here. Type IL_4EEEventCommandType01 function blocks are used here. The Command block IL_4EEEventCommandType02 is not used. It is reserved for exceptional instances in which Command blocks in programs that have already been compiled should be registered during program runtime.

The EventCommand blocks are linked to the consumer Device blocks, with one EventCommand block required per consumer. In this instance, therefore, four EventCommand blocks are used for the conveyor belt and three axes. They specify the "Standby" energy mode for the consumers if no workpieces are awaiting processing. The blocks are linked to the sensor for this reason. If no workpieces are awaiting processing, the sensor activates the EventCommand blocks, i.e., in this instance, they transmit the command "Switch to Standby" to the consumers.

If workpieces are awaiting processing, the four consumers have to be turned back on. Since, however, the four EventCommand blocks that already exist can only specify a precise energy mode (namely "Standby"), four other EventCommand blocks are necessary. These serve to specify the energy mode "Operational" for the four consumers, the conveyor belt and the X-, Y and Z-axes, when workpieces are awaiting processing. To implement this, they are also linked to the sensor and are activated by it when workpieces begin approaching again.

It should also be possible to turn the three handling axes on and off independently of the conveyor belt and lighting. This is easily implemented with

Configuring the EnergyManagement function blocks

the existing architecture. The blocks for turning the three axes on and off are linked to a switch, for a total of six EventCommand blocks being linked to the switch. If the axes should be turned off, the switch activates the three EventCommand blocks for switching off the axes, and then the other three blocks for turning them on. This results in the following architecture for the Handling module:

Configuring the EnergyManagement function blocks

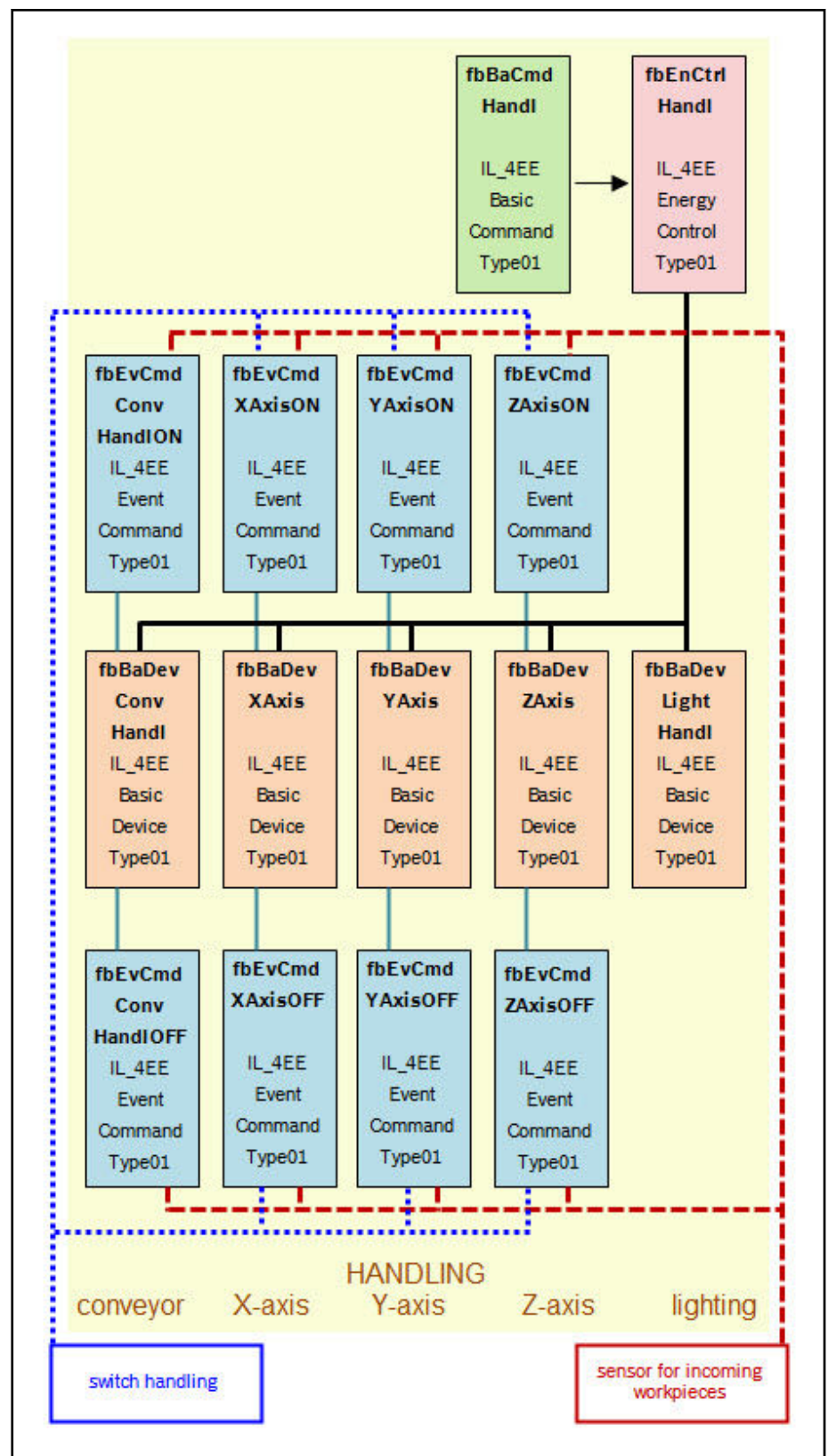


Fig. 5-6: Handling module with sensor and switch

Configuring the EnergyManagement function blocks

5.3.5 Press

The Press station has to be represented by function blocks. Here it should also be possible to turn the entire Press module on and off, or specify dead times. Another BasicCommand block is needed for specification. Device blocks are used to connect the consumers to the architecture. As in Handling, an EnergyControl block is required to link the BasicCommand and BasicDevice blocks together:

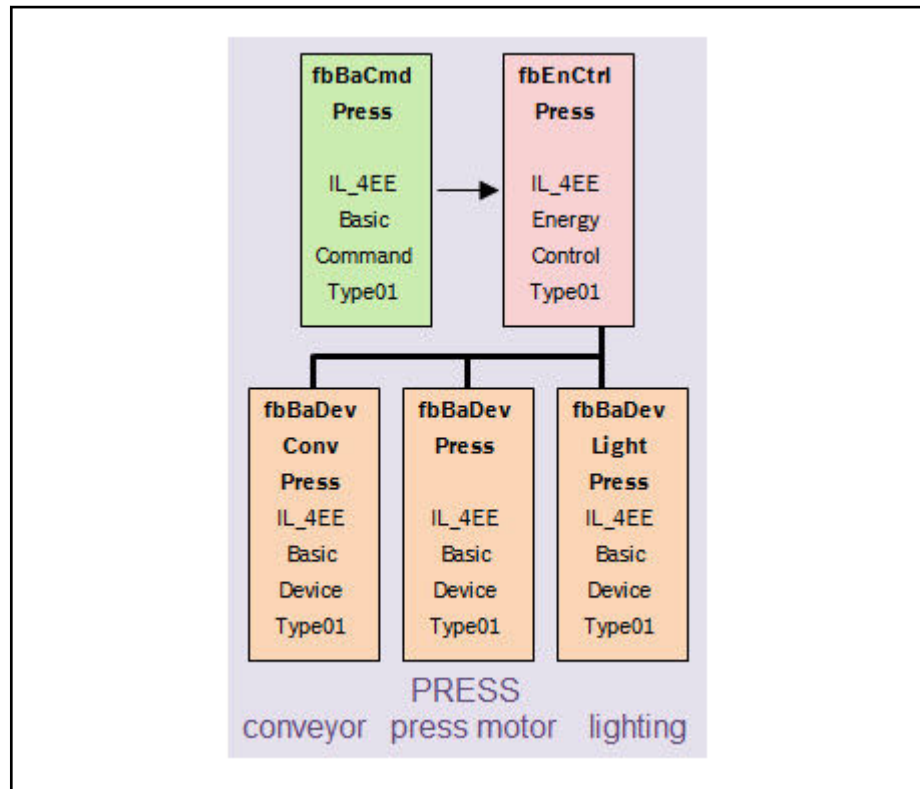


Fig. 5-7: Press module - Basic structure

With the circuitry shown in [fig. 5-7 "Press module - Basic structure" on page 42](#), it is possible to turn all Press consumers on and off, or specify dead times.

Now the sensor integrated for Handling should also be linked to the Press EnergyManagement. If no workpieces are awaiting processing, the press motor and press conveyor belt should be turned off. Two EventCommand blocks are required for the necessary specification of a precise energy mode for individual consumers (here: "Standby"). If no workpieces are awaiting processing, the sensor activates these function blocks and the press motor and conveyor belt are turned off. Only the lighting at the Press should remain on in this instance in order to be able to inspect the station.

If workpieces are awaiting processing, the press motor and press conveyor belt have to be turned back on. Two more EventCommand blocks are added for this that are also linked to the sensor and specify the "Operational" energy mode. If this sensor registers incoming workpieces, both blocks are activated and the press motor and conveyor belt are turned on.

It should also be possible with the Press module to turn the press motor on and off independently from the conveyor belt and lighting. The two existing EventCommand blocks for this purpose are simply connected to a switch. If the press motor should be turned off, the switch activates the EventCom-

Configuring the EnergyManagement function blocks

mand block for turning off the motor, and the EventCommand block for turning it on if it should be turned on.

This results in the following circuit diagram for the Press module:

Configuring the EnergyManagement function blocks

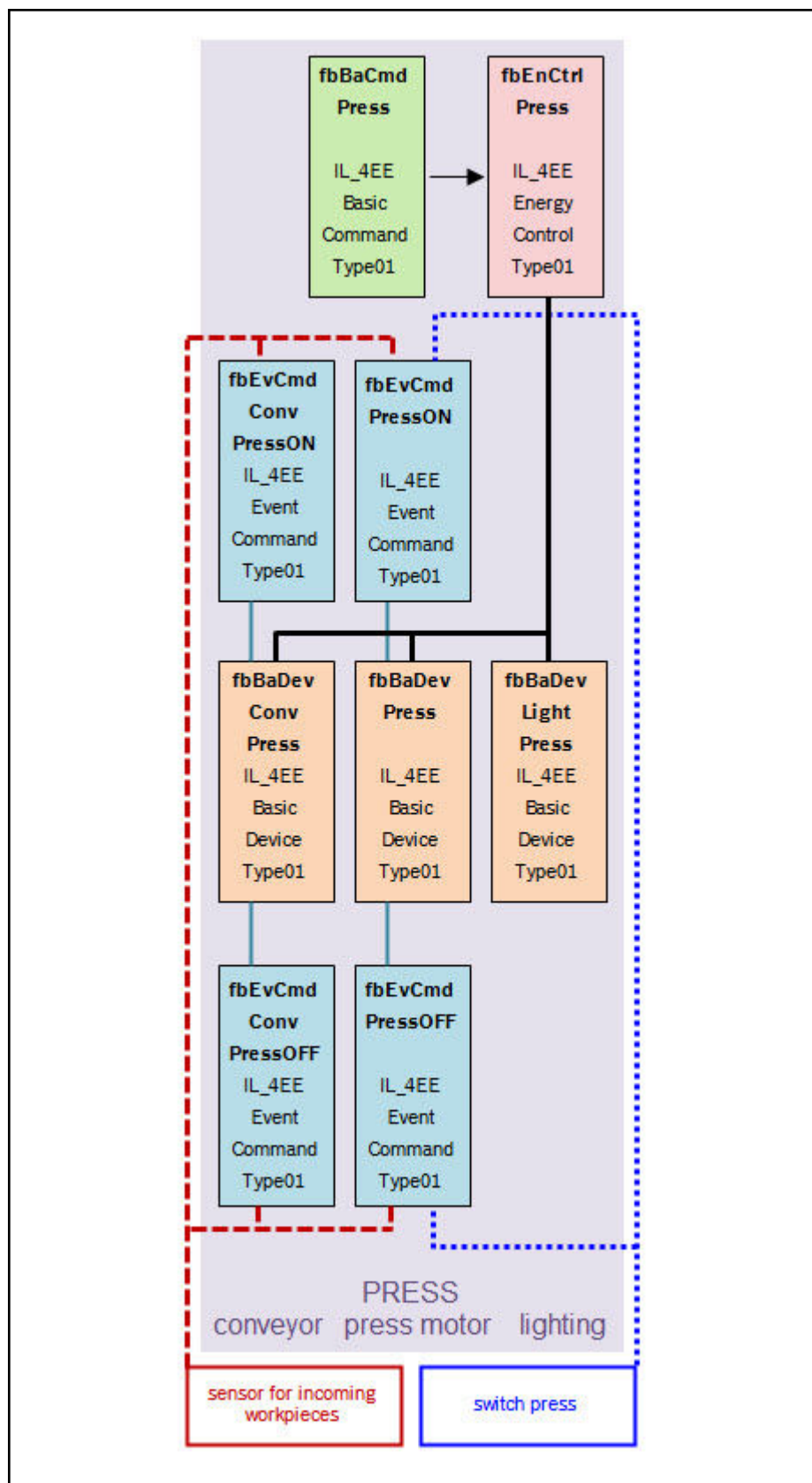


Fig. 5-8: Press module with sensor and switch

Configuring the EnergyManagement function blocks

Both modules can now be activated by a sensor and a switch. However, since only one sensor is available for the sensor inputs, both sensors will be replaced by a single sensor in the next step.

It should be possible to turn the entire machine (i.e., Handling and Press modules) on and off, or specify dead times for the machine. This requires a BasicCommand block. Connecting this block to the existing circuitry requires an EnergyControl block. It is wired to the existing EnergyControl blocks for both modules. For the resulting cascading, the BasicCommand block that specifies the commands for the entire machine has a higher priority than both BasicCommand blocks for each module. This means executing its commands will take precedence. If, for example, the energy mode "Standby" is specified in the higher BasicCommand block, but the operating mode "Operational" is specified in both BasicCommand blocks, all consumers will switch to "Standby" operating mode. The specifications of the two lower-priority BasicCommand blocks are thus ignored if there are higher-priority specifications.

This results in the following arrangement and linking of the function blocks:

Configuring the EnergyManagement function blocks

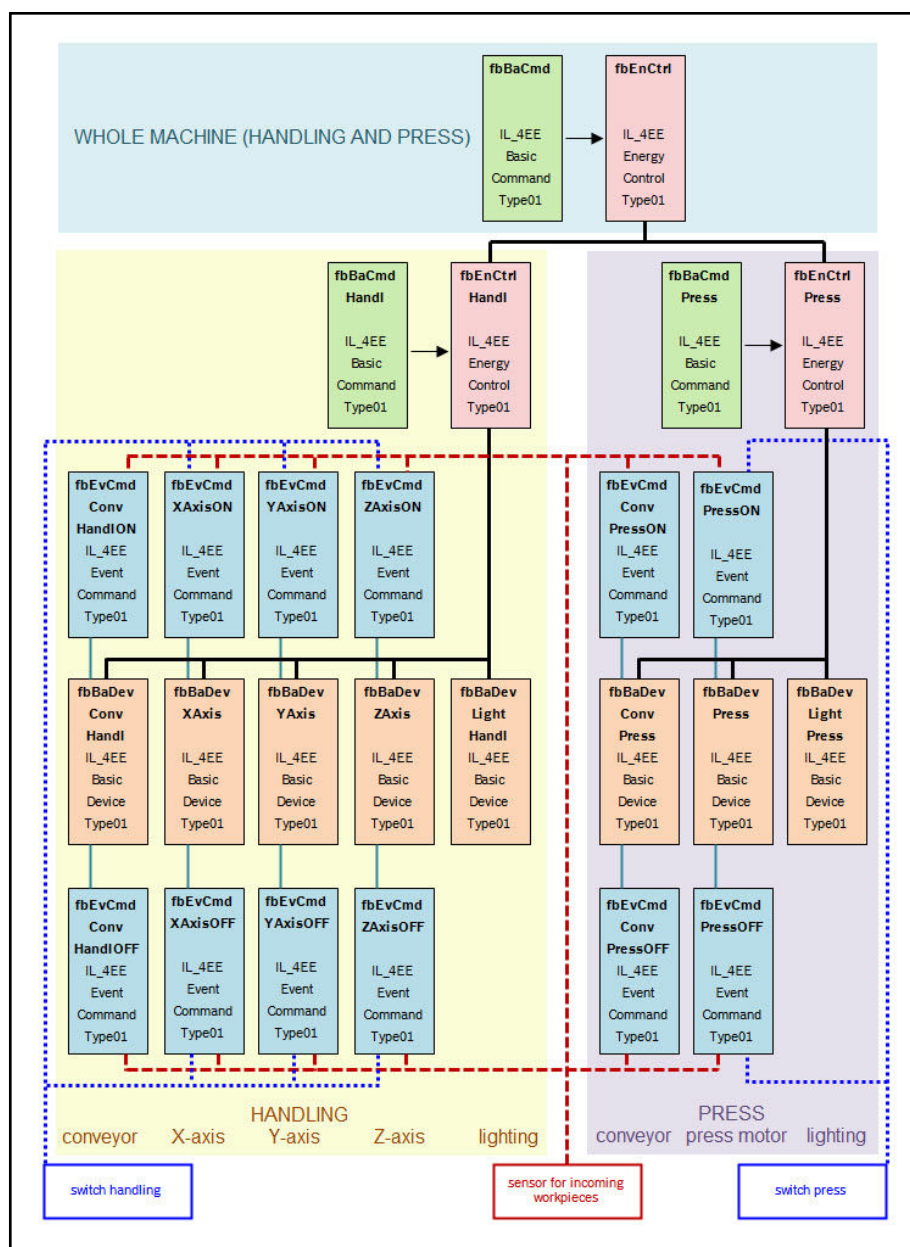


Fig. 5-9: Circuitry of function blocks for representing the entire machine

The operating situations described above can now be realized with the developed architecture through specifications from the function blocks:

- **Normal operation:** All consumers are in operation. "fbBaCmd" specifies energy mode "Operational" for all consumers with the "mSetDirectCommand" method.
- **Turn off all consumers or set to dead time:** All consumers should be turned off. "fbBaCmd" specifies energy mode "Standby" for all consumers with the "mSetDirectCommand" method, or a precise dead time is entered with the "mSetPauseCommand" method. If WakeUp mode is selected, the operational energy mode is automatically switched just before dead time has elapsed so that this mode is in effect when dead time has expired.
- **Single station operation (Handling or Press):** Individual stations can be operated with the two BasicCommand blocks "fbBaCmdHandl" and "fbBaCmdPress". First, the "mResetCommand" method must be called

Configuring the EnergyManagement function blocks

for "fbBaCmd" so that the current specification from this block is reset. This is required since the "fbBaCmd" block has a higher priority than the two blocks "fbBaCmdHandl" and "fbBaCmdPress". Without calling this method, the specifications of the two subordinate blocks would be ignored. After calling the method, the specifications from the "fbBaCmdHandl" and "fbBaCmdPress" can now be entered accordingly for the individual stations. This is done, as previously mentioned, with the "mSetDirectCommand" (specify fixed energy modes) or "mSetPauseCommand" (specify dead times) methods.

- **Machine standstill with light on:** The previously mentioned sensor shows when no workpieces are awaiting processing. If this is the case, the three BasicCommand blocks first reset the current specifications with the mResetCommand method so that the specifications from the EventCommand blocks are not ignored. Then all six available EventCommand blocks can specify the energy mode "Standby"/"Ready for Power" with the mSetEventCommand methods and both lamps remain on.
- **Transporting the workpiece without assembly:** The lamps and conveyor belts should remain on in this instance, while the three axes for Handling and Press should be turned off. To do this, the three BasicCommand blocks first reset the current specifications with the mResetCommand method. Then the four EventCommand blocks for the X-, Y- and Z-axes and for the Press can specify the energy mode "Standby"/"Ready for Power" with the mSetEventCommand method.

6 Realizing the EnergyMonitoring

6.1 Overview

The example scenario is realized using object-oriented programming. The different types of function blocks correspond to classes. IndraWorks is used as development environment. The IEC programming language Structured Text (ST) is used for programming. The "4EE_SmartAutomation" project is contained in an installation below the IndraWorks folder "...\\IndraWorks\\Projects\\4EE\\". The Excel templates for project planning and configuration of the 4EE EnergyManagement function blocks are also contained in this folder. All inserted devices, required libraries (→ library manager), folders, Global Variable Lists, programs, actions, function blocks, visualizations and the task configuration are contained in the project tree.

6.2 Declarations and nesting of the function blocks

The nesting of the function blocks is realized via the declaration. For the function block declaration, the declaration part of the IL_4EE program is used. The basic declaration principle is explained using the example of the EnergyMonitoring function block for the Y-axis of the Handling module:

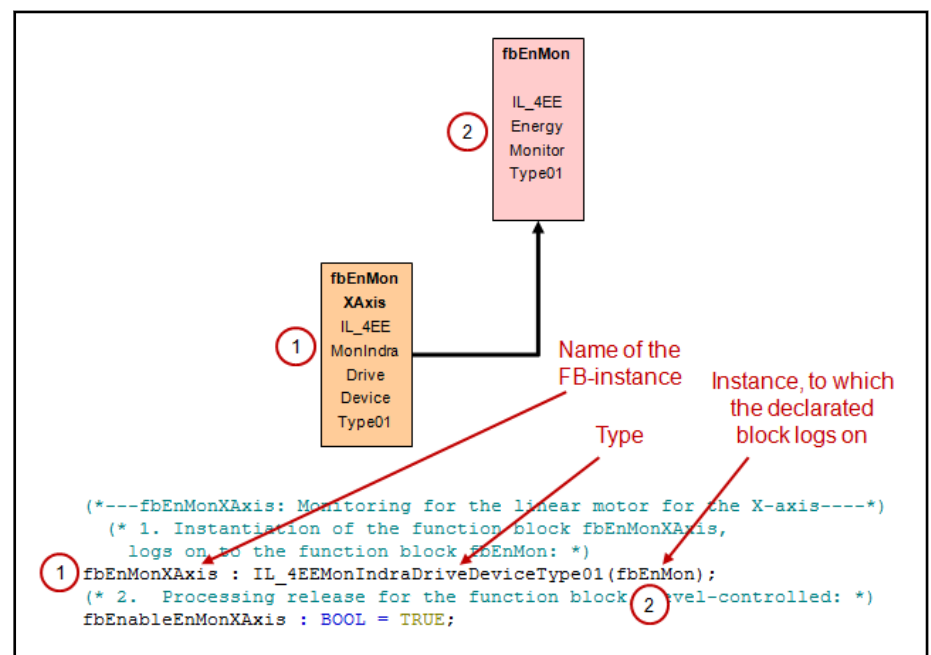


Fig. 6-1: Declaring the type IL_4EEMonIndraDriveDeviceType01 function block "fbEnMonXAxis"

Make sure that a block for which another should be registered already has to be declared. In this example, therefore, the EnergyMonitor block "fbEnMon" has to have been declared before the function block "fbEnMonXAxis". Setting the variable "bEnableEnMonXAxis" to TRUE enables the function block.

At the end of the declaration part, the BOOL data type variable "bInit" is declared:

Program:

```

(*Variable that makes sure that the function blocks are configured
only once, is set to TRUE at the first run, see below*)
bInit : BOOL;
  
```

Realizing the EnergyMonitoring

END_VAR

This variable is necessary so that the action "actConfig" for configuring the blocks only has to be performed one time. To ensure this, the following instruction was initially entered in the implementation part of the IL_4EE program:

Program:

```
IF NOT bInit THEN //if bInit=FALSE (only at the first run)
  actConfig(); //all function blocks were configured...
  bInit:= TRUE; (*bInit is set to TRUE to make sure that the function
  blocks were configured only once*)
END_IF
```

This ensures that the actConfig action only has to be performed once, i.e., the blocks are only configured once during the first run.

6.3 Configuring the EnergyMonitoring

6.3.1 Overview

Executing the "actConfig" action to configure all function blocks was already included as described above. This action contains various EnergyManagement specifications that are to be explained below.

6.3.2 Configuring the EnergyMonitor block

The EnergyMonitoring blocks are configured via their inputs. Only the SampleTime has to be configured here for the EnergyMonitor block. This SampleTime determines the cycle time with which the EnergyMonitor block reads the energy counters for the MonitoringDevice blocks and calculates the resulting output power. The default value is 1s.

A shorter time can be configured in order to update the output values more frequently. However, the SampleTime should be at least five times greater than the task cycle time with which the function block is called.

A longer SampleTime is recommended above all when using device blocks that read the energy counters for the consumers (e.g., IL_4EEMonIndraDriveDeviceType01), since this also increases parameter read operation timeout.

Program:

```
(*---fbEnMon: Monitoring of the whole machine---*)
fbEnMon(Enable:= bEnableEnMon, SampleTime:= T#1S, InOperation=> , Error=> , ErrorID=> , ErrorIdent=> );
```

6.3.3 Configuring the HMIDataCollector block

The function block IL_4EEMonHMIDataCollectorType01 does not need to be configured.

6.3.4 Configuring the MonitoringDevice blocks

The configuration of the IL_4EEMonBasicDeviceType01 is explained using the example of the fbEnMonConvHandl instance for the Handling module belt. The DeviceName input sets the name of the consumer for the HMI display. The TotalConsumption input is only needed for the total energy counter on the machine. If this input is TRUE, the sum of all other consumers is subtracted from this value and the result is shown as the "Remaining Consumption". This was not implemented in the sample project.

The UseEnergyOdometer input allows switching between totaling output values and using an energy counter. The value FALSE was selected here for the function block fbEnMonConvHandl, since the power consumption was

Realizing the EnergyMonitoring

specified with the `mSetPowerConsumption` method. This method is also called in the configuration with a value of 100 W. When the machine is in operation, the `mSetConsumerActive` method notifies the block whether or not the consumer is on or off. Alternatively, updated output values could also be specified using the `mSetPowerConsumption` method.

The other inputs, `DeviceType`, `EnergyType` and `GroupInGrouping1-4`, are used to assign the consumer to consumer groups. The belt motor is a drive that is preset as the enumeration value `DevTypeDrive`. The `EnergyType` input allows differentiating between various energy forms, such as electrical and mechanical energy. The consumers can be divided into further, user-defined groups with the inputs `GroupInGrouping1-4`. In this example, the Handling (Value 1) and Press (Value 2) modules are differentiated by the input `GroupInGrouping1`.

Program:

```
(*---fbEnMonConvHandl: Monitoring of the conveyor of the handling module---*)
fbEnMonConvHandl(Enable:= bEnableEnMonConvHandl, DeviceName:= 'Conveyor Handl.',
  InOperation=> , Error=> , ErrorID=> , ErrorIdent=> , TotalConsumption:= FALSE,
  UseEnergyOdometer:= FALSE, DeviceType:= DEV_TYPE_DRIVE, EnergyType:= E_TYPE_ELECTRICAL,
  GroupInGrouping1:= 1, GroupInGrouping2:= , GroupInGrouping3:= , GroupInGrouping4:= );
(*---Set the average power consumption of the conveyor of the handling module [W]---*)
fbEnMonConvHandl.mSetPowerConsumption(PowerConsumptionValue := 100);
```



In regard to assigning consumers into groups, the current HMI controls only support division into drives and units with the input `DeviceType`. The inputs `EnergyType` and `GroupInGrouping1-4` are currently not evaluated.

The configuration of the function block `IL_4EEMonIndraDriveDeviceType01` is explained using the example of the `fbEnMonXAxis` instance for Handling X-axis. The `DeviceName` input sets the name of the consumer for the HMI display. The inputs `BusMaster` and `SercosAdr` set the drive whose energy counter is to be read.

The consumer can be divided into further, user-defined groups with the inputs `GroupInGrouping1-4`. The groups `DeviceType` and `EnergyType` are preset internally to drive and electrical energy.

Program:

```
(*---fbEnMonXAxis: Monitoring for the linear motor for the X-axis---*)
fbEnMonXAxis(Enable:= bEnableEnMonXAxis, DeviceName:= 'X-Axis',
  InOperation=> , Error=> , ErrorID=> , ErrorIdent=> ,
  BusMaster:= IL_BUSMASTER_0, SercosAdr:= 1, GroupInGrouping1:= 1,
  GroupInGrouping2:= , GroupInGrouping3:= , GroupInGrouping4:= );
```

6.4 Connecting the consumers to EnergyMonitoring

The IndraDrive axes were already connected during configuration. Here only the function block has to be called cyclically. For the belt drives and lighting, the function blocks must be notified whether or not the consumers are on or off. For this reason, the appropriate variables for the onboard outputs are evaluated in the `actMonitoring` action. These have the value `TRUE` when the consumer is on.

Program:

```
(*---Set in the fbEnMonConvHandl, if the conveyor of the handling module is activated---*)
IL_4EE.fbEnMonConvHandl.mSetConsumerActive(ConsumerActive:= bConvHandl);
(*---Set in the fbEnMonLightHandl, if the lighting of the handling module is activated---*)
IL_4EE.fbEnMonLightHandl.mSetConsumerActive(ConsumerActive:= bLightHandl);
(*---Set in the fbEnMonConvPress, if the conveyor of the press module is activated---*)
IL_4EE.fbEnMonConvPress.mSetConsumerActive(ConsumerActive:= bConvPress);
(*---Set in the fbEnMonLightPress, if the lighting of the press module is activated---*)
IL_4EE.fbEnMonLightPress.mSetConsumerActive(ConsumerActive:= bLightPress);
```

6.5 Using commissioning visualization

A commissioning visualization was created for the functionality example. The visualization OverviewEnergyMonitoring shows an overview of the function block architecture. Clicking on the blocks switches to the single view of each instance.

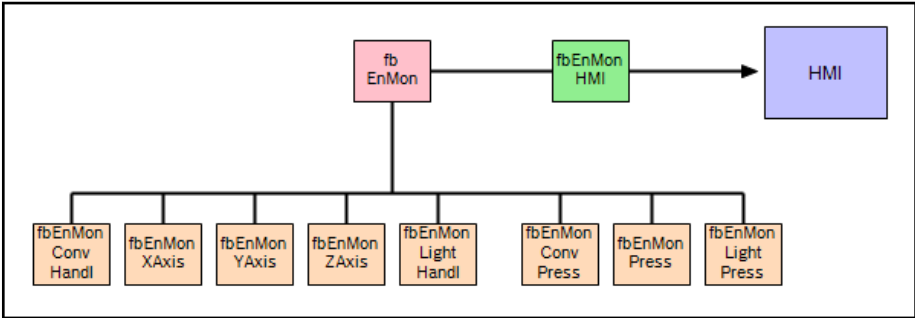


Fig. 6-2: Overview of Commissioning Visualization for EnergyMonitoring

Built-in visualization elements are available that can be used to operate and display the modes of the blocks:

Function block/structure	Visualization
IL_4EEEnergyMonitorType01	EnergyMonitorType01
IL_4EEMonHMIDataCollectorType01	MonHMIDataCollectorType01
IL_4EEMonBasicDeviceType01	MonBasicDeviceType01
IL_4EEMonIndraDriveDeviceType01	MonIndraDriveDeviceType01

Tab. 6-1: Available visualization elements for the function blocks

These visualizations were first added to the project under the menu item **Application ► Data Transfer**. This allows a frame to be inserted in the single view of each block via **Tools ► Frame**. The desired visualization was selected with the Frame option. This can then be linked to the corresponding instance that is to be displayed with the added visualization element via **Properties ► Referenced Visualization**. The necessary elements can be added to each visualization in this manner.

EnergyMonitorType01: The currently configured SampleTime is shown in addition to the Enable inputs and the InOperation and Error outputs.

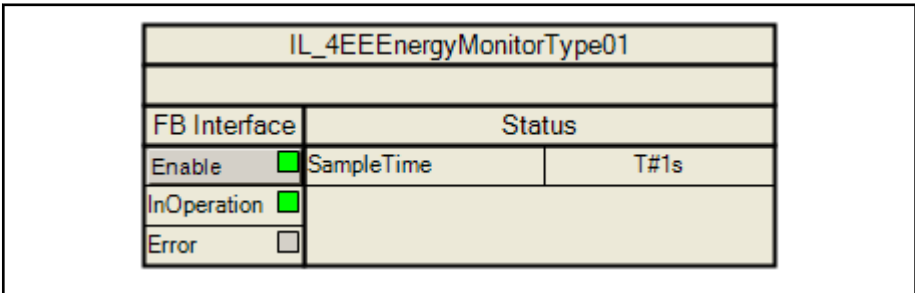


Fig. 6-3: Visualization of EnergyMonitor function block

MonHMIDataCollectorType01: The inputs for starting, stopping and canceling measurement can be operated via the visualization.

Realizing the EnergyMonitoring

IL_4EEMonHMIDataCollectorType01		
Conveyor Handl.		
FB Interface	Inputs	
Enable ☒	StartEMeas	FALSE
InOperation ☒	StopEMeas	FALSE
Error ☐	AbortEMeas	FALSE

Fig. 6-4: Visualization of MonHMIDataCollector function block

MonBasicDeviceType01: The buttons mSetEnergyOdometer, mSetPowerConsumption and mConsumerActive can be used to execute each method with the adjacent, provided values.

IL_4EEMonBasicDeviceType01		
Conveyor Handl.		
FB Interface	Status	
Enable	☒ UseEnergyOdometer	FALSE
InOperation	☒ Energy Odometer Value	3443
Error	☐	
Methods		
mSetEnergyOdometer	0	
mSetPowerConsumption	0	
mConsumerActive	FALSE	

Fig. 6-5: Visualization of MonBasicDevice function block

MonIndraDriveDeviceType01: The mSetAdditiveEnergyOdometer button can be used to preset the additional power consumption with the adjacent, provided value.

IL_4EEMonIndraDriveDeviceType01		
X-Axis		
FB Interface	Status	
Enable ☒	Energy Odometer Value	0
InOperation ☒	TimeoutCount	0
Error ☐		
Methods		
mSetAdditiveEnergyOdometer	0	

Fig. 6-6: Visualization of MonIndraDriveDevice function block

7 Realizing EnergyControl

7.1 Overview

The example scenario is realized using object-oriented programming. The different types of function blocks correspond to classes. IndraWorks is used as development environment. The IEC programming language Structured Text (ST) is used for programming.

- The MTX project "4EE_SmartAutomation" as a functional example is contained in an installation in the IndraWorks folder "...\\IndraWorks\\Projects\\4EE\\". The Excel templates for configuring the 4EE EnergyManagement function blocks are also contained in this folder.
- The XLC/MLC project "4EE_SmartAutomation" as a functional example is contained in the "...\\Addons\\4EE\\" directory of the IndraWorks MLC installation CD. The Excel templates for configuring the 4EE EnergyManagement function blocks are also contained in this folder.

All inserted devices, required libraries (→ Library Manager), folders, Global Variable Lists, programs, actions, function blocks, visualizations and the task configuration are contained in the project tree.

7.2 Declarations and nesting of the function blocks

The nesting of the function blocks is realized via the declaration. For the function block declaration, the declaration part of the IL_4EE program is used. The basic principle of declaration is explained using the EnergyControl function block for the Handling station:

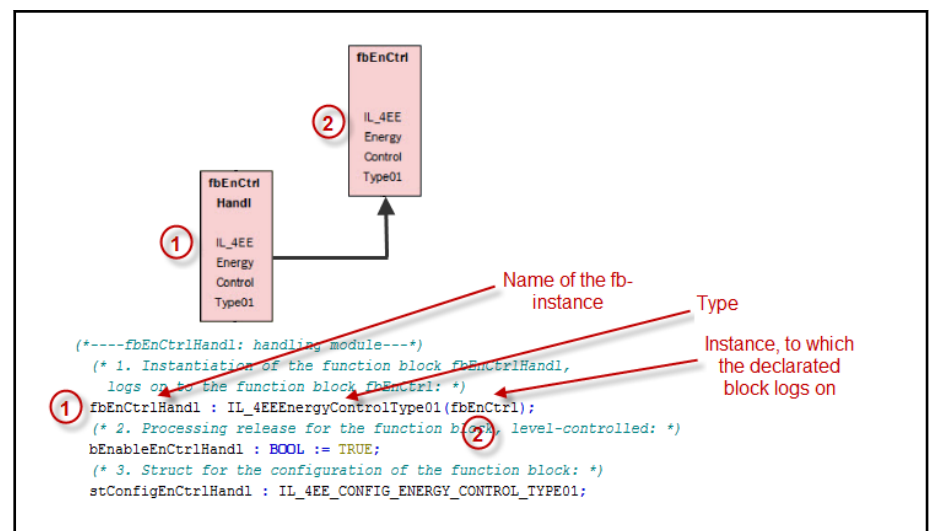


Fig. 7-1: Declaration of function block `fbEnCtrlHandl` of class `IL_4EEEnergyControlType01`

Make sure that a block for which another should be registered already has to be declared. In this example, therefore, the EnergyControl block "fbEnCtrl" has to have been declared before the function block "fbEnCtrlHandl". Setting the variable "bEnableEnCtrlHandl" to TRUE enables the function block. The "stConfigEnCtrlHandl" structure serves to configure the instance.

At the end of the declaration part, the BOOL data type variable "bInit" is declared:

Realizing EnergyControl

Sample Code

```
(*Variable that makes sure that the function blocks are configured
only once, is set to TRUE at the first run, see below*)
bInit : BOOL;
```

```
END_VAR
```

Fig. 7-2: Variable "bInit"

This variable is necessary so that the action "actConfig" for configuring the blocks only has to be performed one time. To ensure this, the following instruction was initially entered in the implementation part of the IL_4EE program:

Program:

```
IF NOT bInit THEN //if bInit=FALSE (only at the first run)
    actConfig(); //all function blocks were configured...
    bInit:= TRUE; (*bInit is set to TRUE to make sure that the function
blocks were configured only once*)
END_IF
```

This ensures that the "actConfig" action only has to be performed once, i.e., the blocks are only configured once during the first run.

After configuration, the function blocks are called in the implementation part of the "IL_4EE" program. Variables continue to be set for the block inputs. The basic principle is explained again using the EnergyControl function block for the Handling station:

Program:

```
(*----fbEnCtrlHandl: handling module----*)
fbEnCtrlHandl(Enable:= bEnableEnCtrlHandl, Config:= stConfigEnCtrlHandl,
    InOperation=> , Error=> , ErrorID=> , ErrorIdent=> );
```

Enable/disable the function block with the "bEnableEnCtrlHandl" variable via the "Enable" input. Assign the input/output "Config" with the "stConfigEnCtrlHandl" configuration structure.

The other blocks are called according to the same principle, which can be found in the source code of the "IL_4EE" program.

7.3 Configuring the EnergyControl

7.3.1 Overview

Executing the "actConfig" action to configure all function blocks was already included as described above. This action contains various EnergyManagement specifications that are to be explained below.

7.3.2 Configuring the EnergyControl function blocks

Configuring the EnergyControl blocks (see [fig. 7-3 "EnergyControl block" on page 57](#)) is explained using the "fbEnCtrlHandl" block. This is responsible for the EnergyManagement of the Handling station. Specify the settings for the other EnergyControl blocks in the same manner. Essentially, concrete values are assigned to the "stConfigEnCtrlHandl (IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01)" structure instance when configuring the attributes.

Realizing EnergyControl

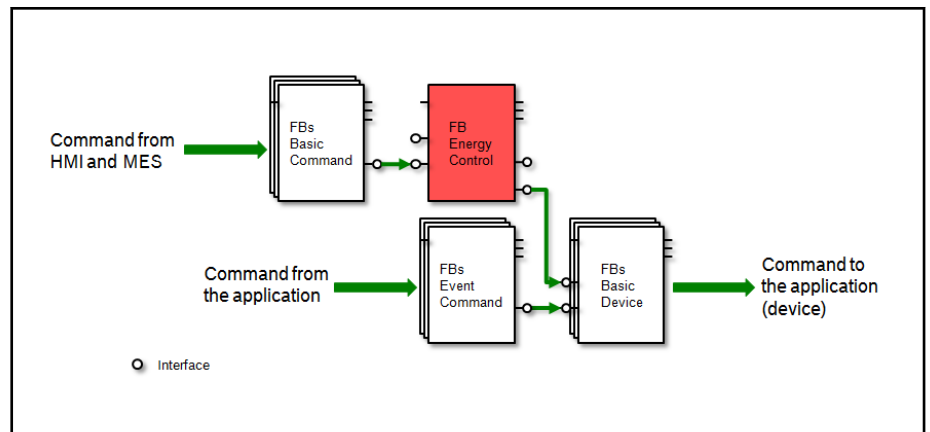


Fig. 7-3: EnergyControl block

The structure `IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01` contains the necessary data for configuring the EnergyControl block. The `EMConfig` attribute is an array of `IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01` structures for setting the energy modes.

The `EMAssignment` attribute is an array of `IL_4EE_EM_ASSIGNMENT_TYPE01` structures that serves to assign the energy modes for the EnergyControl block to the energy modes of the higher EnergyControl function block. In this example, the array has three elements based on the three possible energy modes.

The screenshot shows the STEP 7 LAD editor with the 'IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01' structure selected. The left pane shows the project tree with the 'Config' folder expanded. The right pane shows the structure definition table. Red circles and arrows highlight specific elements: (1) points to the 'IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01' structure in the project tree, (2) points to the 'IL_4EE_EM_ASSIGNMENT_TYPE01' structure in the project tree, and (3) points to the 'EMAssignment' attribute in the structure definition table.

Name	Datentyp	Geerbt von	Adresse	Initialwert	Kommentar
ControlName	STRING(16)				Name of the EnergyControl-Fb for the HMI (machine-name or module-name)
EMNameOp	STRING(16)				Name of the operational mode for the HMI
PowerConsumptionEMOp	REAL			0	Mean power consumption [W] in operational mode without process
TimeMinStayEMOp	TIME			TIME#0ms	Minimum time in the operational mode
EMConfig	ARRAY [1..NO_OF_EM_PER_ENERGY_CONTROL] OF IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01				Array of struct for the configuration of the energy-modes
EMAssignment	ARRAY [0..NO_OF_EM_PER_ENERGY_CONTROL] OF IL_4EE_EM_ASSIGNMENT_TYPE01				Correlation of the energy modes of the EnergyControl-FB and a superordinate EnergyControl-FB
StatusOverride	BOOL			FALSE	Override for the interpretation of the CurrentEM has to be set by the user with mSetState FALSE: Status is interpreted by the status of the DeviceFB

Name	Datentyp	Geerbt von	Adresse	Initialwert	Kommentar
EMValid	BOOL			FALSE	TRUE: The energy mode is considered by the energy management
EMIdent	USINT				ID of the energy mode (hexadecimal coded)
EMName	STRING(16)				Name of the energy mode for the HMI (e.g. StandBy)
TimeToEMx	TIME				Needed time [s] for the change from energy mode Operational to the configured energy mode
TimeToEMOp	TIME				Needed time [s] for the change from the configured energy mode to energy mode Operational
TimeMinStay	TIME				Maximum residence time [s] in the energy mode
TimeMaxStay	TIME				Minimum residence time [s] in the energy mode
PowerConsumption	REAL				Mean power consumption [W] in the configured energy mode
RequeToEMx	REAL				Energy required [Ws] for the change from energy mode Operational to the configured energy mode
RequeToEMOp	REAL				Energy required [Ws] for the change from the configured energy mode to energy mode Operational

Name	Datentyp	Geerbt von	Adresse	Initialwert	Kommentar
ControlEM	USINT		16#FF		Energy mode of the superordinate EnergyControl-FB
TargetEM	USINT		16#FF		Assigned energy mode of the subordinated FB

Fig. 7-4: Parameters for the `IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01` structure, and `IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01` and `IL_4EE_EM_ASSIGNMENT_TYPE01` substructures

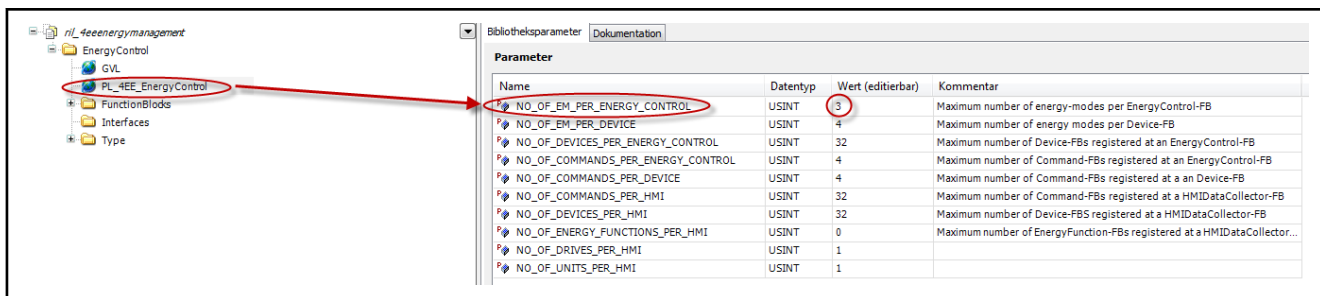
Realizing EnergyControl

The following Excel template serves to support the configuration of the EnergyControl blocks. The EnergyControl block has to be added and completed for each module.

Energy Control block				
Instance name	fbEnCtrl			
Instance name of structure	stConfigEnCtrl			
Description	Control of both modules			
Energy Mode	Config. data	Value	Type	Description
EM_Operational	stConfigEnCtrl.ControlName	'Whole machine'	STRING(16)	Name of the function block - machine or module
	stConfigEnCtrl.StatusOverride	FALSE	BOOL	Feedback of the energy mode, two options:
	stConfigEnCtrl.EMNameOp	'Operational'	STRING(16)	Name of the operating energy mode
	stConfigEnCtrl.PowerConsumption	1200	REAL	Average power consumption [W] in operating energy mode
	stConfigEnCtrl.TimeMinStayEMOp	T#0S	TIME	Minimum dwell time [s] in operating energy mode
EM_1	stConfigEnCtrl.EMAssignment[0].ControlEM	16#FF	USINT	Assignment to machine energy mode. ControlEM: ID of the target energy mode of the superordinate EnergyControl-FB, in hexadecimal code
	stConfigEnCtrl.EMAssignment[0].TargetEM	16#FF	USINT	Assignment to machine energy mode. TargetEM: ID of the associated target energy mode of the subordinate FB, in hexadecimal code
	stConfigEnCtrl.EMConfig[1].EMValid	TRUE	BOOL	TRUE: The energy mode will be configured
	stConfigEnCtrl.EMConfig[1].EMName	'Ready for Power'	STRING(16)	Name of the energy mode EM 1
	stConfigEnCtrl.EMConfig[1].EMIdent	16#40	USINT	ID of the energy mode EM 1, hexadecimal code
EM_2	stConfigEnCtrl.EMConfig[1].PowerConsumption	100	REAL	Average power consumption [W] in energy mode EM 1
	stConfigEnCtrl.EMConfig[1].ReqEToEMOp	15000	REAL	Energy consumption [Ws] for the change from energy mode EM 1 to
	stConfigEnCtrl.EMConfig[1].ReqEToEMx	3000	REAL	Energy consumption [Ws] for the change from operating energy mode to
	stConfigEnCtrl.EMConfig[1].TimeMaxStay	T#0S	TIME	Maximum dwell time [s] in energy mode EM 1, will be ignored when the
	stConfigEnCtrl.EMConfig[1].TimeMinStay	T#0S	TIME	Minimum dwell time [s] in energy mode EM 1
	stConfigEnCtrl.EMConfig[1].TimeToEMOp	T#10S	TIME	Time [s] required for the change from energy mode EM 1 to operating
	stConfigEnCtrl.EMConfig[1].TimeToEMx	T#3S	TIME	Time [s] required for the change from operating energy mode to energy
	stConfigEnCtrl.EMAssignment[1].ControlEM	16#40	USINT	Assignment to machine energy mode. ControlEM: ID of the target energy mode of the superordinate EnergyControl-FB, in hexadecimal code
	stConfigEnCtrl.EMAssignment[1].TargetEM	16#40	USINT	Assignment to machine energy mode. TargetEM: ID of the associated target energy mode of the subordinate FB, in hexadecimal code
	stConfigEnCtrl.EMConfig[2].EMValid	TRUE	BOOL	TRUE: The energy mode will be configured
	stConfigEnCtrl.EMConfig[2].EMName	'Standby'	STRING(16)	Name of the energy mode EM 2
	stConfigEnCtrl.EMConfig[2].EMIdent	16#20	USINT	ID of the energy mode EM 2, hexadecimal code
	stConfigEnCtrl.EMConfig[2].PowerConsumption	50	REAL	Average power consumption [W] in energy mode EM 2
	stConfigEnCtrl.EMConfig[2].ReqEToEMOp	20000	REAL	Energy consumption [Ws] for the change from energy mode EM 2 to
	stConfigEnCtrl.EMConfig[2].ReqEToEMx	5000	REAL	Energy consumption [Ws] for the change from operating energy mode to
	stConfigEnCtrl.EMConfig[2].TimeMaxStay	T#0S	TIME	Maximum dwell time [s] in energy mode EM 2, will be ignored when the
	stConfigEnCtrl.EMConfig[2].TimeMinStay	T#0S	TIME	Minimum dwell time [s] in energy mode EM 2
	stConfigEnCtrl.EMConfig[2].TimeToEMOp	T#12S	TIME	Time [s] required for the change from energy mode EM 2 to operating
	stConfigEnCtrl.EMConfig[2].TimeToEMx	T#5S	TIME	Time [s] required for the change from operating energy mode to energy
	stConfigEnCtrl.EMAssignment[2].ControlEM	16#20	USINT	Assignment to machine energy mode. ControlEM: ID of the target energy mode of the superordinate EnergyControl-FB, in hexadecimal code
	stConfigEnCtrl.EMAssignment[2].TargetEM	16#20	USINT	Assignment to machine energy mode. TargetEM: ID of the associated target energy mode of the subordinate FB, in hexadecimal code

Fig. 7-5: Excel template to support the configuration of the EnergyControl block

The maximum number of configurable energy modes can be determined through the constant NO_OF_EM_PER_ENERGY_CONTROL, with the default value being 8. In this example, the maximum three different energy modes are needed ("Standby", "Ready for Power", "Operational"), so NO_OF_EM_PER_ENERGY_CONTROL can be set to three in the Library Manager under "PL_4EE_EnergyControl".



Parameter				
Name	Datentyp	Wert (editierbar)	Kommentar	
NO_OF_EM_PER_ENERGY_CONTROL	USINT	3	Maximum number of energy-modes per EnergyControl-FB	
NO_OF_EM_PER_DEVICE	USINT	4	Maximum number of energy modes per Device-FB	
NO_OF_DEVICES_PER_ENERGY_CONTROL	USINT	32	Maximum number of Device-FBs registered at an EnergyControl-FB	
NO_OF_COMMANDS_PER_ENERGY_CONTROL	USINT	4	Maximum number of Command-FBs registered at an EnergyControl-FB	
NO_OF_COMMANDS_PER_DEVICE	USINT	4	Maximum number of Command-FBs registered at a Device-FB	
NO_OF_COMMANDS_PER_HMI	USINT	32	Maximum number of Command-FBs registered at a HMIDataCollector-FB	
NO_OF_DEVICES_PER_HMI	USINT	32	Maximum number of Device-FBs registered at a HMIDataCollector-FB	
NO_OF_ENERGY_FUNCTIONS_PER_HMI	USINT	0	Maximum number of EnergyFunction-FBs registered at a HMIDataCollector...	
NO_OF_DRIVES_PER_HMI	USINT	1		
NO_OF_UNITS_PER_HMI	USINT	1		

Fig. 7-6: Constant NO_OF_EM_PER_ENERGY_CONTROL

First set the name of the machine/module controlled by the EnergyControl function blocks (ControlName). The name later appears when visualizing the blocks. Here "Handling module" is selected as the name, whereby a string of up to 16 characters is permitted.

Realizing EnergyControl

Next, specify the type of feedback for the current energy mode in the Energy-Control block (StatusOverride). There are two options here: TRUE means that the user returns the current energy mode via the mSetState method. Specifying FALSE, however, determines the current energy mode from the energy modes of the subordinate BasicDevice function blocks. This results in the following source code for the "fbEnCtrlHandl" block:

Program:

```
(*-----fbEnCtrlHandl: handling module-----*)
(*name of the function block - machine oder module*)
stConfigEnCtrlHandl.ControlName:= 'Handling module';
(*feedback of the energy mode to the function block fbEnCtrlHandl,
two options:
- TRUE: the user returns the current energy mode by using the
method mSetState
- FALSE: the current energy mode is determined from the energy modes
of the modes of the devices*)
stConfigEnCtrlHandl.StatusOverride:= FALSE;
```

Now describe the possible energy modes. First set the design parameters for the operational energy mode. The following figure shows some important values:

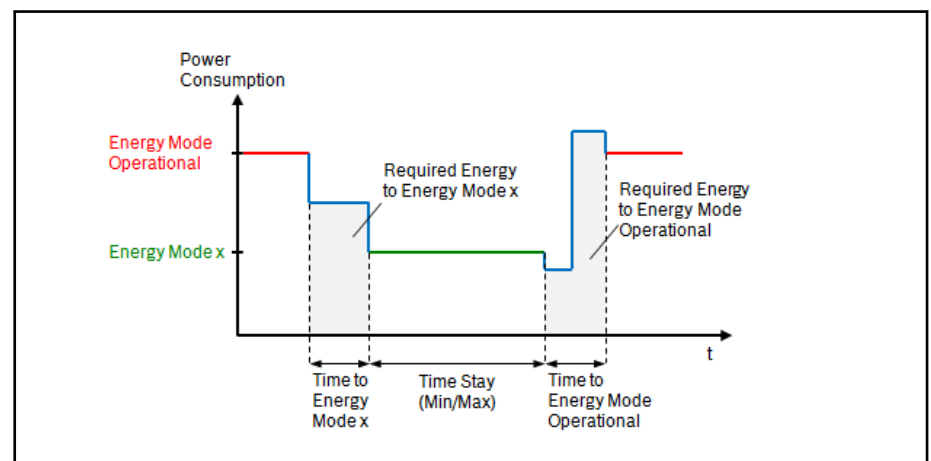


Fig. 7-7: Selected configuration data for an IL_4EEEnergyControlType01 energy mode

First set the design parameters for the operational energy mode. "Operational" is the name of the energy mode selected in this example. Now specify the average energy consumption in watts (here 1200 W). At this value, be sure to specify the average consumption for a machine at a standstill without process forces. If the average consumption during operation were specified, a practical comparison with the other energy modes would not be possible.

Setting the minimum dwell time in the operational energy mode (TimeMinStayEMOp) can be used to protect the switching contacts. All commands are ignored during this time. This time was set to 0 seconds for the "fbEnCtrlHandl" function block.

Program:

```
(*operating energy mode "Operational"*)
(*name of the operating energy mode*)
stConfigEnCtrlHandl.EMNameOp:= 'Operational';
(*average power consumption [W] in operating energy mode*)
stConfigEnCtrlHandl.PowerConsumptionEMOp := 1200;
(*minimum dwell time [s] in operating energy mode*)
stConfigEnCtrlHandl.TimeMinStayEMOp:= T#0S;
```

Now other energy modes can be defined, which in this example are the energy modes "Ready for Power" and "Standby". There are thus a total of three

Realizing EnergyControl

possible energy modes. First the settings for the "Ready for Power" energy mode are explained.

First determine whether the energy mode should be configured or not even included by setting the relevant attribute EMValid to TRUE (configured) or FALSE (not included). "Ready for Power" should be configured in the sample project. Now set a name for the energy mode and a coded hexadecimal ID as the characteristic number for the mode. In the example, "Ready for Power" is selected for the name, and "16#40" is set for the characteristic number.

Continue by specifying the average output power and energy consumption, as well as the time needed to switch from the energy mode being configured to the operational energy mode and back. The minimum dwell time in the energy mode being configured can, as explained above, be used to protect the switching contacts. Also specify the maximum dwell time after which the machine automatically switches to the operational energy mode. If this attribute is 0, it will be ignored and no automatic switching occurs. This produces the following source code:

Program:

```
(*-> energy mode "Ready for Power"*)
(*TRUE: the energy mode will be configured, FALSE: the energy mode
will be ignored*)
stConfigEnCtrlHandl.EMConfig[1].EMValid:= TRUE;
(*name of the energy mode , z. B. „Standby"*)
stConfigEnCtrlHandl.EMConfig[1].EMName:= 'Ready for Power';
(*ID of the energy mode, hexadecimal code*)
stConfigEnCtrlHandl.EMConfig[1].EMIdent:= 16#40;
(*average power consumption [W] in energy mode "Ready for Power"*)
stConfigEnCtrlHandl.EMConfig[1].PowerConsumption:= 100;
(*energy consumption [Ws] for the change from energy mode "Ready
for Power" to operating energy mode*)
stConfigEnCtrlHandl.EMConfig[1].RequETOEMOp:= 15000;
(*energy consumption [Ws] for the change from operating energy
mode to energy mode "Ready for Power"*)
stConfigEnCtrlHandl.EMConfig[1].RequETOEMx:= 3000;
(*maximum dwell time [s] in energy mode "Ready for Power",
will be ignored when the variable is set to 0*)
stConfigEnCtrlHandl.EMConfig[1].TimeMaxStay:= T#0S;
(*minimum dwell time [s] in energy mode "Ready for Power"*)
stConfigEnCtrlHandl.EMConfig[1].TimeMinStay:= T#0S;
(*time [s] required for the change from energy mode "Ready
for Power" to operating energy mode*)
stConfigEnCtrlHandl.EMConfig[1].TimeToEMOp:= T#10S;
(*time [s] required for the change from operating energy mode to
energy mode "Ready for Power"*)
stConfigEnCtrlHandl.EMConfig[1].TimeToEMx:= T#3S;
```

In this example, there are two other energy modes in addition to the operational energy mode - this is set by NO_OF_EM_PER_ENERGY_CONTROL. Accordingly, there are two elements in the array with Array Indexes 1 and 2. Index "[1]" is used for the specifications for the "Ready for Power" energy mode.

Index "[2]" is used for the specifications for the "Standby" energy mode. Implement the specifications for this energy mode in the same manner:

Program:

```
(*-> energy mode "Standby"*)
(* TRUE: the energy mode will be configured,
FALSE: the energymode will be ignored *)
stConfigEnCtrlHandl.EMConfig[2].EMValid:= TRUE;
(*name of the energy mode , z. B. „Standby"*)
stConfigEnCtrlHandl.EMConfig[2].EMName:= 'Standby';
(*ID of the energy mode, hexadecimal code*)
stConfigEnCtrlHandl.EMConfig[2].EMIdent:= 16#20;
(*average power consumption [W] in energy mode "Standby"*)
stConfigEnCtrlHandl.EMConfig[2].PowerConsumption:= 50;
(*energy consumption [Ws] for the change from energy mode
"Standby" to operating energy mode*)
stConfigEnCtrlHandl.EMConfig[2].RequETOEMOp:= 20000;
(*energy consumption [Ws] for the change from operating energy
```

```
mode to energy mode "Ready for Power"*)
stConfigEnCtrlHandl.EMConfig[2].RequETOEMx:= 5000;
(*maximum dwell time [s] in energy mode "Standby", will
be ignored when the variable is set to 0*)
stConfigEnCtrlHandl.EMConfig[2].TimeMaxStay:= T#0S;
(*minimum dwell time [s] in energy mode "Standby"*)
stConfigEnCtrlHandl.EMConfig[2].TimeMinStay:= T#0S;
(*time [s] required for the change from energy mode "Standby" to
operating energy mode*)
stConfigEnCtrlHandl.EMConfig[2].TimeToEMOp:= T#12S;
(*time [s] required for the change from operating energy mode to
energy mode "Standby"*)
stConfigEnCtrlHandl.EMConfig[2].TimeToEMx:= T#5S;
```

For EnergyControl blocks with a higher EnergyControl block, each energy mode (TargetEM) has to be assigned an energy mode for the higher EnergyControl block (ControlEM). The sample scenario contains such cascading. "fbEnCtrl" (EnergyControl block for both stations) overrides "fbEnCtrlHandl" (EnergyControl block for the Handling station). The "fbEnCtrlHandl" energy modes have the same coded hexadecimal IDs as the "fbEnCtrl" energy modes. This results in the following assignment:

Program:

```
(*Assignment to machine energy modes*)
(*ControlEM: ID of the target energy mode of the superordinate
EnergyControl-FB, in hexadecimal code*)
(*TargetEM: ID of the associated target energy mode of the
subordinate FB, in hexadecimal code*)
stConfigEnCtrlHandl.EMAssignment[0].ControlEM:= 16#FF;//Operational
stConfigEnCtrlHandl.EMAssignment[0].TargetEM:= 16#FF;//Operational
stConfigEnCtrlHandl.EMAssignment[1].ControlEM:= 16#40//R. for Power
stConfigEnCtrlHandl.EMAssignment[1].TargetEM:= 16#40//R. for Power
stConfigEnCtrlHandl.EMAssignment[2].ControlEM:= 16#20;//Standby
stConfigEnCtrlHandl.EMAssignment[2].TargetEM:= 16#20;//Standby
```

Configure the remaining EnergyControl blocks ("fbEnCtrl" and "fbEnCtrlPress") in the same manner. Make sure the EMAssignment substructure is not needed for the higher "fbEnCtrl" EnergyControl block, since there is no EnergyControl block that is higher.

7.3.3 Configuring the command function blocks

This section explains the configuration parameters for the Command function blocks.

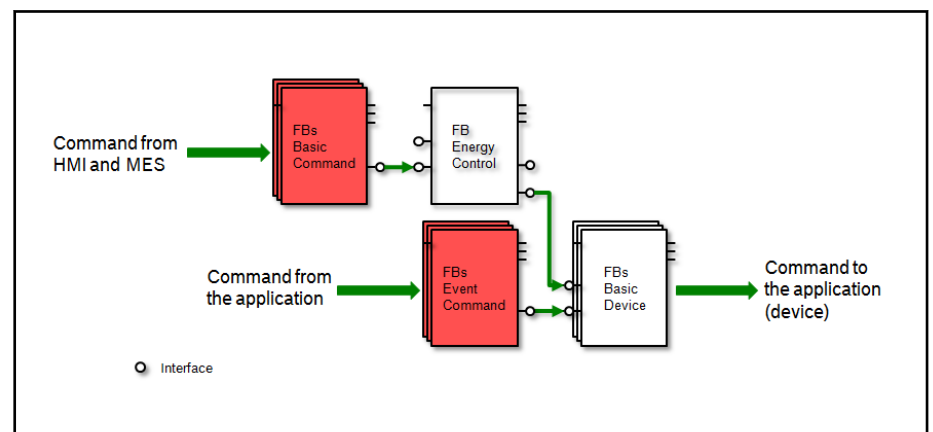


Fig. 7-8: BasicCommand and EventCommand blocks

The structure IL_4EE_CONFIG_BASIC_COMMAND_TYPE01 contains the necessary data for configuring the BasicCommand block. The necessary data for the EventCommand block can be configured via the structure IL_4EE_CONFIG_BASIC_COMMAND_TYPE01.

Realizing EnergyControl

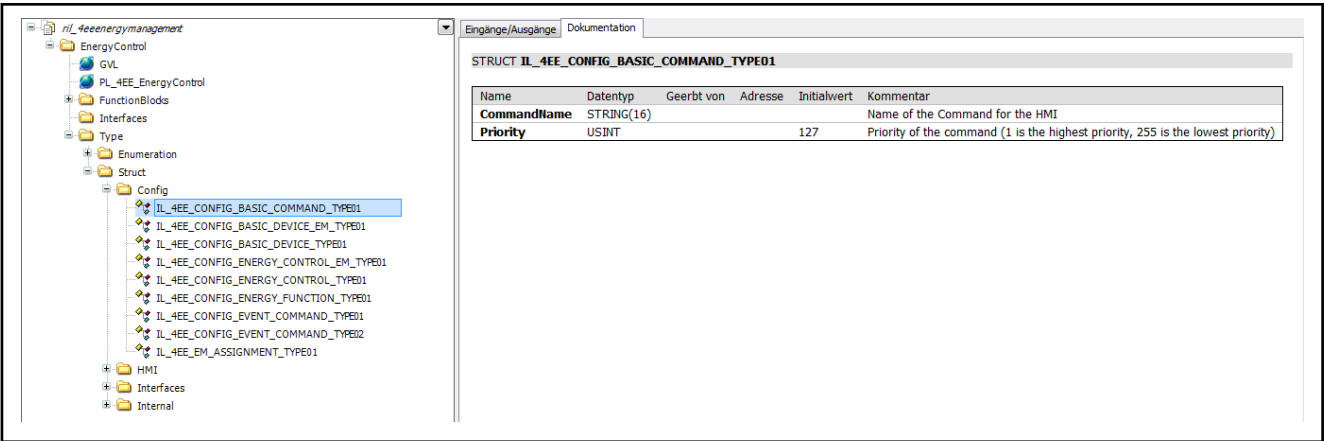


Fig. 7-9: Parameters for the IL_4EE_CONFIG_BASIC_COMMAND_TYPE01 structure

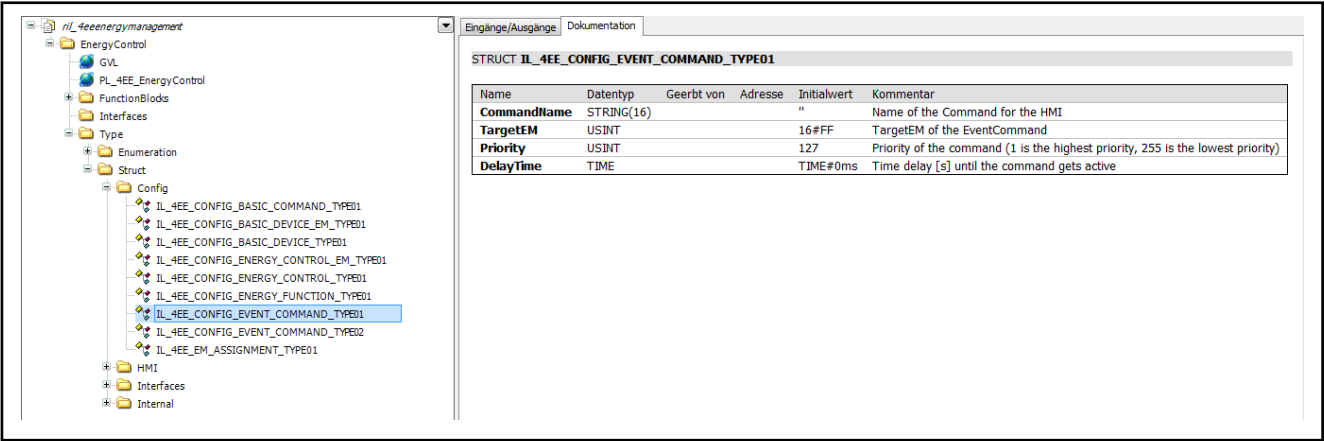


Fig. 7-10: Parameters for the IL_4EE_CONFIG_BASIC_COMMAND_TYPE01 structure

The following Excel template serves to support the configuration of the Basic-Command blocks. A BasicCommand block has to be added and completed for the external interface (HMI entries, machine operating panel, MES, etc.).

Realizing EnergyControl

Command block				
Instance name	fbBaCmd			
Instance name of structure	stConfigBaCmd			
Description	BasicCommand for both module			
Config. data	Value	Typ	Description	Note
stConfigBaCmd.CommandName	'Whole machine'	STRING (16)	Name of the command	
stConfigBaCmd.Priority	30	USINT	Priority of the command , 1: highest priority, 255: lowest priority	
Command block				
Instance name	fbBaCmdHandl			
Instance name of structure	stConfigBaCmdHandl			
Description	BasicCommand for handling module			
Config. data	Value	Typ	Description	Note
stConfigBaCmdHandl.CommandName	'Handling module'	STRING (16)	Name of the command	
stConfigBaCmdHandl.Priority	40	USINT	Priority of the command , 1: highest priority, 255: lowest priority	
Command block				
Instance name	fbBaCmdPress			
Instance name of structure	stConfigBaCmdPress			
Description	BasicCommand for press module			
Config. data	Value	Typ	Description	Note
stConfigBaCmdPress.CommandName	'Press module'	STRING (16)	Name of the command	
stConfigBaCmdPress.Priority	40	USINT	Priority of the command , 1: highest priority, 255: lowest priority	

Fig. 7-11: Excel template for the BasicCommand block

The following Excel template serves to support the configuration of the EventCommand blocks. List all events in the machine that pertain to energy in this table.

Event blocks		
Instance name	fbEvCmdConvHandlOff	fbEvCmdConvHandlOn
Description	Conveyor of the handling module off	Conveyor of the handling module on
Name of the Command	'ConvHandlOFF'	'ConvHandlON'
Delay Time [s]	T#5s	T#5s
Priority	60	50
Target Energy Mode	16#20	16#FF

Fig. 7-12: Excel template for the EventCommand block

Prioritization: The priority of the Command function blocks determines which commands take precedence and which are ignored when other commands are given.

The following requirements are realized in the functional example with prioritization:

1. It should be possible to specify energy modes for the entire machine (i.e., for all consumers) in "fbBaCmd". If specifications are made here, the specifications in all other Command blocks should be ignored.
2. It should be possible to enter specifications for all modules in "fbBaCmdHandl" and "fbBaCmdPress". Consumer commands should be ignored when specifications exist for the entire module.

Realizing EnergyControl

3. Specifications can be made for individual consumers in the 12 EventCommand blocks. However, the commands from the BasicCommand blocks described under 1 and 2 should take precedence.

Therefore the following Command blocks exist for which the described prioritization should be realized:

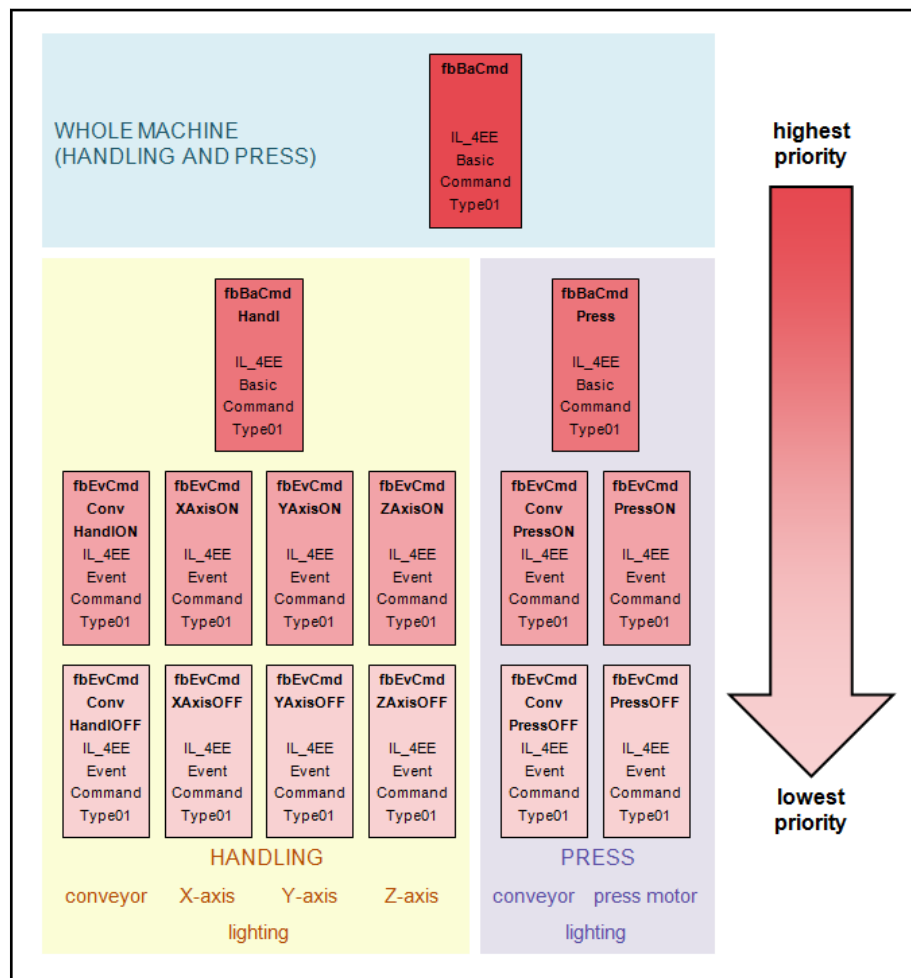


Fig. 7-13: Command block prioritization

If function block A has a higher priority than function block B, then the commands from A take precedence and the commands from function block B are ignored if both sets of commands exist simultaneously. In the sample scenario, the "fbBaCmd" has a higher priority than the "fbBaCmdHandl" and "fbBaCmdPress" blocks so that the commands from "fbBaCmd" take precedence and those from "fbBaCmdHandl" and "fbBaCmdPress" are ignored. The EventCommand blocks have the lowest priority. A number between 1 and 255 can be set for the priority, whereby 1 has the highest priority and 255 the lowest. The following condition thus has to be met:

$\text{Priority}(\text{fbBaCmd}) < \text{Priority}(\text{fbBaCmdHandl}/\text{fbBaCmdPress}) < \text{Priority}(\text{EvCmd function blocks})$

This is why the priority for "fbBaCmd" was set to 30, and at 40 for the other two BasicCommand blocks, "fbBaCmdHandl" and "fbBaCmdPress". A priority of 50 (ON) and 60 (OFF) was selected for the EventCommand blocks.

BasicCommand function blocks: This example contains, as previously described, three BasicCommand function blocks: "fbBaCmd" for both stations, "fbBaCmdHandl" for the Handling station and "fbBaCmdPress" for the Press

station. In essence, two attributes are set: The name of the command and the priority of the BasicCommand block:

Program:

```
(*----fbBaCmd for both modules----*)
(* name of the command *)
stConfigBaCmd.CommandName:= 'Whole machine';
(*priority of the command, 1:highest priority, 255:lowest priority*)
stConfigBaCmd.Priority:= 30;

(*----fbBaCmdHandl: handling module---*)
(* name of the command *)
stConfigBaCmdHandl.CommandName:= 'Handling module';
(*priority of the command, 1:highest priority, 255:lowest priority*)
stConfigBaCmdHandl.Priority:= 40;

(*----fbBaCmdPress: press module---*)
(* name of the command *)
stConfigBaCmdPress.CommandName:= 'Press module';
(*priority of the command, 1:highest priority, 255:lowest priority*)
stConfigBaCmdPress.Priority:= 40;
```

EventCommand function blocks: The sample scenario has 12 different EventCommand blocks, with no EventCommand blocks needed for lighting both stations. Configuration is explained using the "fbEvCmdConvHandlOFF" block (EventCommand block for switching off the Handling station conveyor belt).

First set a name for the command, in this case "ConvHandlOFF". This name later appears when visualizing the function blocks. The "DelayTime" is recommended if the command should be rescinded within this time without the EnergyManagement reacting. In the sample scenario, this can be beneficial if the sensor indicates a temporary lack of workpieces, but other workpieces are soon to arrive for processing. In this instance, the delay time means the machine does not stop regardless of the temporary lack of workpieces, and processing can continue uninterrupted. In this instance, the delay is set to 10 seconds. Also set a priority for the command. Lastly, set the target energy mode that the EventCommand block should specify. Unlike the BasicCommand blocks, this was already set for the EventCommand blocks during configuration. In this instance, this is "Standby" (ID=16#20). This results in the following source code for "fbEvCmdConvHandlOFF":

Program:

```
(*----fbEvCmdConvHandlOFF: conveyor of the handling module OFF----*)
(* name of the command *)
stConfigEvCmdConvHandlOFF.CommandName:= 'ConvHandlOFF';
(*delay time [s] until the command becomes active*)
stConfigEvCmdConvHandlOFF.DelayTime:= T#10S;
(*priority of the command, 1:highest priority, 255:lowest priority*)
stConfigEvCmdConvHandlOFF.Priority:= 60;
(*target energy mode of the command, value range: 16#00 - 16#FF*)
stConfigEvCmdConvHandlOFF.TargetEM:= 16#20; //Standby
```

Configure the other EventCommand blocks in the same manner.

7.3.4 Configuring the BasicDevice function blocks

The sample scenario contains a total of eight different consumers. Accordingly eight different BasicDevice function blocks have to be configured. The configuration is explained using the "fbBaDevConvHandl" block (Handling station conveyor belt).

Realizing EnergyControl

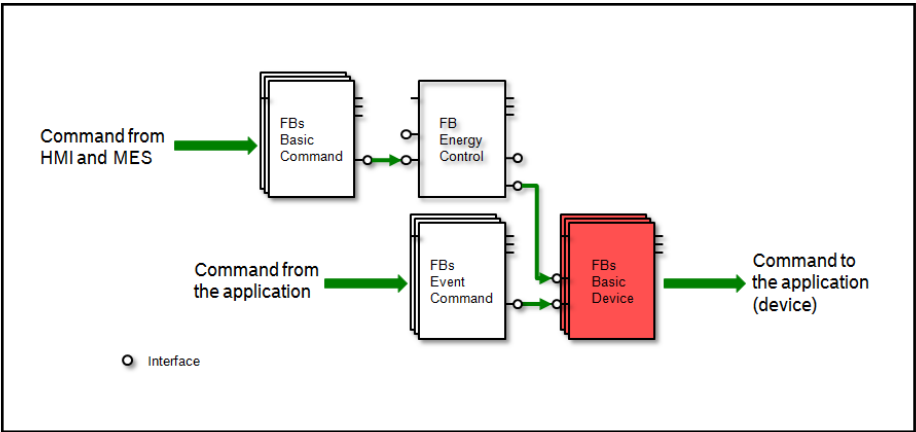


Fig. 7-14: BasicDevice blocks

The necessary data for the BasicDevice block can be configured via the structure IL_4EE_CONFIG_BASIC_DEVICE_TYPE01. The IL_4EE_CONFIG_BASIC_DEVICE_EM_TYPE01 substructure contains the configuration for an energy mode for the block. Assign the energy modes for the BasicDevice block to the energy modes for the higher EnergyControl function block via the IL_4EE_EM_ASSIGNMENT_TYPE01 substructure.

The screenshot shows the configuration structure IL_4EE_CONFIG_BASIC_DEVICE_TYPE01 in a software tool. The left pane shows a tree view of the configuration, with IL_4EE_CONFIG_BASIC_DEVICE_EM_TYPE01 and IL_4EE_EM_ASSIGNMENT_TYPE01 highlighted. The right pane shows the detailed definition of IL_4EE_CONFIG_BASIC_DEVICE_TYPE01.

Name	Datentyp	Geerbt von	Adresse	Initialwert	Kommentar
DeviceName	STRING(16)				Name of the Device for the HMI
EMNameOp	STRING(16)				Name of the operational mode for the HMI
PowerConsumptionEMOp	REAL			0	Mean power consumption [W] in operational mode without process
EMConfig	ARRAY [1..NO_OF_EM_PER_DEVICE] OF IL_4EE_CONFIG_BASIC_DEVICE_EM_TYPE01				Array of struct for the configuration of the energy modes
EMAssignment	ARRAY [0..NO_OF_EM_PER_ENERGY_CONTROL] OF IL_4EE_EM_ASSIGNMENT_TYPE01				Correlation of the energy modes of the Device-FB and the superordinate EnergyControl-FB

Name	Datentyp	Geerbt von	Adresse	Initialwert	Kommentar
EMValid	BOOL			FALSE	TRUE: The energy mode is considered by the energy management
EMIdent	USINT				ID of the energy mode (hexadecimal coded)
EMName	STRING(16)				Name of the energy mode for the HMI
PowerConsumption	REAL				Mean power consumption [W] in the configured energy mode

Name	Datentyp	Geerbt von	Adresse	Initialwert	Kommentar
ControlEM	USINT		16#FF		Energy mode of the superordinate EnergyControl-FB
TargetEM	USINT		16#FF		Assigned energy mode of the subordinated FB

Fig. 7-15: Parameters for the IL_4EE_CONFIG_BASIC_DEVICE_TYPE01 structure

The following Excel template serves to support the configuration of the Basic-Device blocks. Complete the configuration data for each device.

Realizing EnergyControl

Device No	1			
Instance Name	fbBaDevConvHandl			
Instance Name of structure	stConfigBaDevConvHandl			
Description	Conveyor of the handling module			
Config. Data	Config. Data	Value	Type	Description
EM_Operational	stConfigBaDevConvHandl.DeviceName	'Conveyor Handl'	STRING(16)	Name of the device
	stConfigBaDevConvHandl.EMNameOp	'Operational'	STRING(16)	Name of the operating energy mode
	stConfigBaDevConvHandl.PowerConsumptionEMOp	1000	REAL	Average power consumption [W] in operating energy mode
	stConfigBaDevConvHandl.EMAssignment[0].ControlEM	16#FF	USINT	Assignment to machine energy mode. ControlEM: ID of the target energy mode of the superordinate EnergyControl-FB, in hexadecimal code
	stConfigBaDevConvHandl.EMAssignment[0].TargetEM	16#FF	USINT	Assignment to machine energy mode. TargetEM: ID of the associated target energy mode of the subordinate FB, in hexadecimal code
EM_1	stConfigBaDevConvHandl.EMConfig[1].EMValid	TRUE	BOOL	TRUE: energy mode EM_1 will be taken into account by the function block.
	stConfigBaDevConvHandl.EMConfig[1].EMName	'Ready for Power'	STRING(16)	Name of the energy mode EM_1
	stConfigBaDevConvHandl.EMConfig[1].EMIdent	16#30	USINT	ID of the energy mode EM_1, hexadecimal code
	stConfigBaDevConvHandl.EMConfig[1].PowerConsumption	100	REAL	Average power consumption [W] in Energy mode EM_1
	stConfigBaDevConvHandl.EMAssignment[1].ControlEM	16#40	USINT	Assignment to machine energy mode. ControlEM: ID of the target energy mode of the superordinate EnergyControl-FB, in hexadecimal code
EM_2	stConfigBaDevConvHandl.EMAssignment[1].TargetEM	16#20	USINT	Assignment to machine energy mode. TargetEM: ID of the associated target energy mode of the subordinate FB, in hexadecimal code
	stConfigBaDevConvHandl.EMConfig[2].EMValid	TRUE	BOOL	TRUE: energy mode EM_2 will be taken into account by the function block.
	stConfigBaDevConvHandl.EMConfig[2].EMName	'Standby'	STRING(16)	Name of the energy mode EM_2
	stConfigBaDevConvHandl.EMConfig[2].EMIdent	16#20	USINT	ID of the energy mode EM_2, hexadecimal code
	stConfigBaDevConvHandl.EMConfig[2].PowerConsumption	50	REAL	Average power consumption [W] in Energy mode EM_2
	stConfigBaDevConvHandl.EMAssignment[2].ControlEM	16#20	USINT	Assignment to machine energy mode. ControlEM: ID of the target energy mode of the superordinate EnergyControl-FB, in hexadecimal code
	stConfigBaDevConvHandl.EMAssignment[2].TargetEM	16#20	USINT	Assignment to machine energy mode. TargetEM: ID of the associated target energy mode of the subordinate FB, in hexadecimal code

Fig. 7-16: Excel template for the BasicDevice block

Edit the NO_OF_EM_PER_DEVICE constant before configuring each device block. It sets the number of energy modes for the device blocks, with the default value at 4. In this example, however, only three different energy modes are possible for the consumers ("Operational", "Ready for Power", "Standby"). The value of the constant can therefore be set to three in the Library Manager under "PL_4EE_EnergyControl".

First set the name of the device, here it is "ConveyorHandling". Now specify the various energy modes - an array is used here again, as with the EnergyControl block. Finally, assign a device energy mode to each energy mode for the EnergyControl block for which the IL_4EEBasicDeviceType01 instance is registered, to produce a table assignment as with the cascading of EnergyControl blocks. Make sure the "Ready for Power" energy mode is not possible for the conveyor belts and the lamps for both modules and there are only the energy modes "Operational" and "Standby", or "On" and "Off". If the BasicCommand function blocks specify the energy mode "Ready for Power", the energy mode "Standby" (16#20) is transmitted to these four consumers as the target energy mode. On the other hand, the energy mode "Standby" is specified for the three Handling axes and the press motor.

Program:

```
(*----fbBaDevConvHandl: conveyor of the handling module----*)
(*name of the device *)
stConfigBaDevConvHandl.DeviceName:= 'Conveyor Handl';

(*operating energy mode "Operational"*)
(*name of the operating energy mode*)
stConfigBaDevConvHandl.EMNameOp:= 'Operational';
(*average power consumption [W] in operating energy mode*)
stConfigBaDevConvHandl.PowerConsumptionEMOp:= 1000;

(*-> energy mode "Ready for Power"*)
(*TRUE: energy mode "Ready for Power" will be taken into account
   by the function block, FALSE: the energy mode will be ignored*)
stConfigBaDevConvHandl.EMConfig[1].EMValid:= TRUE;
(*name of the energy mode , z. B. „Standby“*)
stConfigBaDevConvHandl.EMConfig[1].EMName:= 'Ready for Power';
(*ID of the energy mode, hexadecimal code*)
stConfigBaDevConvHandl.EMConfig[1].EMIdent:= 16#30;
(*average power consumption [W] in energy mode "Ready for Power"*)
stConfigBaDevConvHandl.EMConfig[1].PowerConsumption:= 100;
```

Realizing EnergyControl

```
(*-> energy mode "Standby"*)
(*TRUE: energy mode "Standby" will be taken into account
  by the function block, FALSE: the energy mode will be ignored*)
stConfigBaDevConvHandl.EMConfig[2].EMValid:= TRUE;
(*name of the energy mode *)
stConfigBaDevConvHandl.EMConfig[2].EMName:= 'Standby';
(*ID of the energy mode, hexadecimal code*)
stConfigBaDevConvHandl.EMConfig[2].EMIdent:= 16#20;
(*average power consumption [W] in energy mode "Ready for Power"*)
stConfigBaDevConvHandl.EMConfig[2].PowerConsumption:= 50;

(*Assignment to machine energy modes*)
(*ControlEM: ID of the target energy mode of the superordinate
  EnergyControl-FB, hexadecimal code*)
(*TargetEM: ID of the associated target energy mode of the
  subordinate FB, hexadecimal code*)
stConfigBaDevConvHandl.EMAssignment[0].ControlEM:=16#FF;//Operational
stConfigBaDevConvHandl.EMAssignment[0].TargetEM:=16#FF;//Operational
stConfigBaDevConvHandl.EMAssignment[1].ControlEM:=16#40;//R. for Power
stConfigBaDevConvHandl.EMAssignment[1].TargetEM:=16#20;//R. for Power
stConfigBaDevConvHandl.EMAssignment[2].ControlEM:=16#20;//Standby
stConfigBaDevConvHandl.EMAssignment[2].TargetEM:=16#20;//Standby
```

Configure the other BasicDevice function blocks in the same manner.

7.4 Connecting the consumers to EnergyManagement

7.4.1 Overview

Now establish a connection with the consumers so that the EnergyManagement specifications can be transmitted to them. For this purpose, the IL_4EEApplication program has the action "actInit" that is executed directly after starting the project to initialize the consumers. Each consumer also has an action containing the program parts that have to be run when the target energy mode is changed.

If no actual consumers are available, the sample project can still be used to its full extent. The "VirtualDevice" function block was added for this purpose. It is a timer in which a certain time can be specified with "tSwitchtime". This is the time an actual consumer would need to switch to the specified target energy mode. When this time has elapsed, the variable "OK" is set to TRUE:

Program:

```
IF ZAB.Q = FALSE THEN (*Trigger 1-0-1 again when set time
  running out*)
  ZAB(IN:=FALSE,PT:=tSwitchtime,ET=>ETVar);
  ZAB(IN:=TRUE,PT:=tSwitchtime,ET=>ETVar);
ELSE
  ZAB(); //Timer runs
END_IF

IF ZAB.Q = FALSE THEN
  OK:= TRUE;//OK = TRUE when time is running out
END_IF
```

When the specified "tSwitchtime" has elapsed, an actual consumer would switch to the specified target energy mode. Each BasicDevice block can then report with the mSetState method that the consumer is in the target energy mode and switching was successful. If, for example, no actual X-axis is available and a specific target energy mode, "usiTargetEMXAxis", was specified, an actual consumer with a switching time of four seconds can be simulated as follows:

Program:

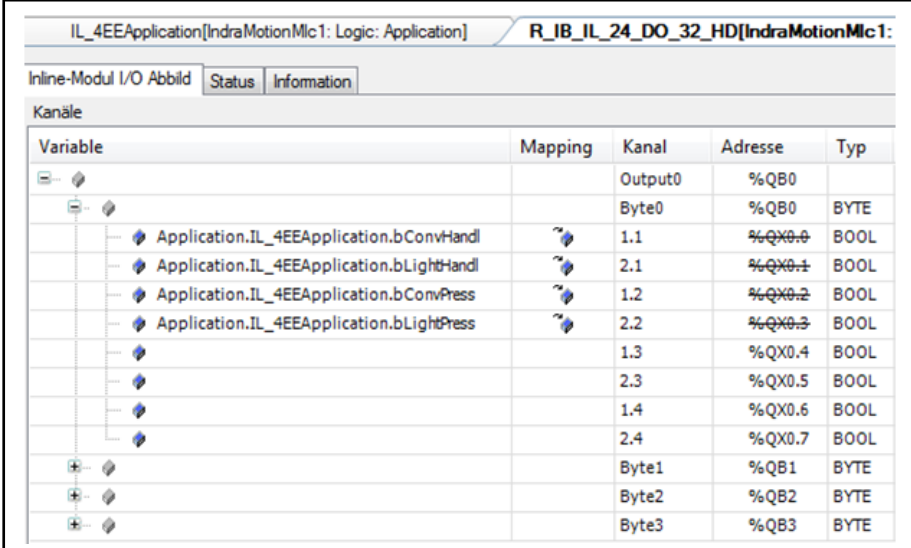
```
ELSE (*Simulating, axis is inactive*)
  VirtualXAxis(tSwitchtime:=T#4S); (*This time needs the
  device for setting the target energy mode*)
IF VirtualXAxis.OK=TRUE THEN
  (*the energy mode has been set, actual energy mode = target
  energy mode*)
  (*After the setting of the energy mode: feedback of the
```

```
current energy mode to the function block fbBaDevXAxis, in
hexadecimal code*)
IL_4EE.fbBaDevXAxis.mSetState(CurrentEM := usiTargetEMXAxis);
END_IF
```

Since no actual conveyor belts and lamps are typically available when testing the program, this consumer was simulated. Active and inactive axes are recognized when starting the program and turned on or simulated accordingly. If no actual axes exist, the axes in the project must be disabled.

7.4.2 Connecting conveyor belts and lighting

The R_IB_IL_24_DO_32_HD module was added under Inline_I/O for this purpose. Its outputs %QX0.0 to %QX0.3 are assigned to the BOOL-type variables "bConvHandl", "bLightHandl", "bConvPress" and "bLightPress". These are declared in the "IL_4EEApplication" program.



Variable	Mapping	Kanal	Adresse	Typ
		Output0	%QB0	
		Byte0	%QB0	BYTE
Application.IL_4EEApplication.bConvHandl		1.1	%QX0.0	BOOL
Application.IL_4EEApplication.bLightHandl		2.1	%QX0.1	BOOL
Application.IL_4EEApplication.bConvPress		1.2	%QX0.2	BOOL
Application.IL_4EEApplication.bLightPress		2.2	%QX0.3	BOOL
		1.3	%QX0.4	BOOL
		2.3	%QX0.5	BOOL
		1.4	%QX0.6	BOOL
		2.4	%QX0.7	BOOL
		Byte1	%QB1	BYTE
		Byte2	%QB2	BYTE
		Byte3	%QB3	BYTE

Fig. 7-17: Variables assigned to the control system outputs

If the operational energy mode for the consumers should be configured, the variables are set to TRUE.

7.4.3 Connecting axes for MLC

AxisInterfaceType01 was used to connect the axes. It was added by pressing the "GAT" button. The X-, Y- and Z-axes for Handling and the press motor were added. Explanations can be found in the documentation "Rexroth IndraMotion Technology Libraries" (DOC-MLC***-TF*LIB**V13-LI01-EN-P). The axes are switched with the "SwitchAxis" function block. The axis number and target energy mode are transmitted to it. After switching, the Boolean variable "OK" is set to TRUE and the new energy mode is output:

Program:

```
FUNCTION_BLOCK SwitchAxis
VAR_INPUT
    usiTargetEM: USINT;
    uiAxisNumber: UINT;
END_VAR
VAR_OUTPUT
    usiCurrentEM: USINT;
    OK : BOOL;
END_VAR
VAR
END_VAR
```

Realizing EnergyControl

An IF instruction differentiates between three instances in the implementation part of the function block. Depending on which target energy mode is transmitted, a series of commands are processed that result in a switch via the AxisInterface. Make sure that it is not possible to directly switch from parameterization mode (corresponds to "Standby") to AH mode (corresponds to "Operational"). Exit parameterization mode first. Switching to AH operating mode is only possible with arAxisCtrl_gb[uiAxisNumber].Admin._OpMode.en := ModeAH after doing so (check with arAxisStatus_gb[uiAxisNumber].Admin.ParamLevelAck). It is also not possible to directly switch from the operational energy mode (AH mode) to "Standby". First switch to AB mode with arAxisCtrl_gb[uiAxisNumber].Admin._OpMode.en := ModeAB, then parameterization mode can be activated with arAxisCtrl_gb[uiAxisNumber].Admin.ParamLevel := TRUE. An IF instruction generally checks whether or not the previous command was executed before the next instruction becomes active, because otherwise the potentially longer duration of switching procedures can result in errors during program execution:

Program:

```

IF (usiTargetEM = 16#FF) THEN (*Operational*)
    arAxisCtrl_gb[uiAxisNumber].Admin.ParamLevel := FALSE;
    IF arAxisStatus_gb[uiAxisNumber].Admin.ParamLevelAck = FALSE THEN
        arAxisCtrl_gb[uiAxisNumber].Admin._OpMode.en:= ModeAH;
    END_IF
    IF arAxisStatus_gb[uiAxisNumber].Admin._OpModeAck = ModeAH THEN
        usiCurrentEM := usiTargetEM;
        OK := TRUE;
    END_IF
ELSIF (usiTargetEM = 16#30) THEN (*Ready for Power*)
    arAxisCtrl_gb[uiAxisNumber].Admin.ParamLevel := FALSE;
    IF arAxisStatus_gb[uiAxisNumber].Admin.ParamLevelAck = FALSE THEN
        arAxisCtrl_gb[uiAxisNumber].Admin._OpMode.en:= ModeAB;
    END_IF
    IF arAxisStatus_gb[uiAxisNumber].Admin._OpModeAck = ModeAB THEN
        usiCurrentEM := usiTargetEM;
        OK := TRUE;
    END_IF
ELSIF (usiTargetEM = 16#20) THEN (*Standby*)
    arAxisCtrl_gb[uiAxisNumber].Admin._OpMode.en := ModeAB;
    IF arAxisStatus_gb[uiAxisNumber].Admin._OpModeAck = ModeAB THEN
        arAxisCtrl_gb[uiAxisNumber].Admin.ParamLevel := TRUE;
    END_IF
    IF arAxisStatus_gb[uiAxisNumber].Admin.ParamLevelAck = TRUE THEN
        usiCurrentEM := usiTargetEM;
        OK := TRUE;
    END_IF
END_IF

```

7.4.4 Connecting the axes for MTX

The SwitchAxis function block was used to connect the axes. Explanations can be found in the documentation "Rexroth IndraMotion MTX 13VRS SPS Interface" (DOC-MTX***-PLC*INT*V13-PR01-EN-P). The axis number and target energy mode are transmitted to the function block. After switching, the Boolean variable "OK" is set to TRUE and the new energy mode is output:

Program:

```

FUNCTION_BLOCK SwitchAxis
    VAR_INPUT
        usiTargetEM: USINT;
        uiAxisNumber: UINT;
    END_VAR
    VAR_OUTPUT
        usiCurrentEM: USINT;
        OK: BOOL;
    END_VAR
    VAR
        bDrvStateAB: BOOL:=FALSE;
        bDrvStateAF: BOOL:=FALSE;
    END_VAR

```

An IF instruction differentiates between three instances in the implementation part of the function block. Depending on which target energy mode is transmitted, a series of commands are processed that result in a switch via the AxisInterface. Make sure that it is not possible to directly switch from BB mode (drive is ready for operation) to AF operating mode (drive in control mode). Exit BB mode first. Once this is done, it is possible to switch to AF operating mode with `stMachine_gb.Ax[uiAxisNumber].NPIq.qAx_DrvOn:=TRUE`. `stMachine_gb.Ax[uiAxisNumber].NPIq.qAx_DrvOn:=FALSE` switches from AF operating mode to AB mode (drive ready for control enable). An IF instruction generally checks whether or not the previous command was executed before the next instruction becomes active, because otherwise the potentially longer duration of switching procedures can result in errors during program execution:

Program:

```
bDrvStateAB:=NOT stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DrvErrClass1 AND
stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DrvPower AND
stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DrvReady AND
NOT stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DrvAct AND
NOT stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DryRun;

bDrvStateAF:=NOT stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DrvErrClass1 AND
stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DrvPower AND
stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DrvReady AND
stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DrvAct AND
NOT stMachine_gb.Ax[uiAxisNumber].NPIi.iAx_DryRun;

IF (usiTargetEM = 16#FF) THEN (*Operational*)
  (*Is drive in AB or already in AF, allow controller enable*)
  IF bDrvStateAB OR bDrvStateAF THEN
    (*Enable controller*)
    stMachine_gb.Ax[uiAxisNumber].NPIq.qAx_DrvOn:=TRUE;
  END_IF
  IF stMachine_gb.Ax[uiAxisNumber].Power.StateAF THEN
    usiCurrentEM := usiTargetEM;
    OK := TRUE;
  END_IF
ELSEIF (usiTargetEM = 16#30) THEN (*Ready for Power*)
  (*Reset control signal for controller enable*)
  stMachine_gb.Ax[uiAxisNumber].NPIq.qAx_DrvOn:=FALSE;
  IF stMachine_gb.Ax[uiAxisNumber].Power.StateAB THEN
    usiCurrentEM := usiTargetEM;
    OK := TRUE;
  END_IF
ELSEIF (usiTargetEM = 16#20) THEN (*Standby*)
  (*Reset control signal for controller enable*)
  stMachine_gb.Ax[uiAxisNumber].NPIq.qAx_DrvOn:=FALSE;
  IF stMachine_gb.Ax[uiAxisNumber].Power.StateAB THEN
    usiCurrentEM := usiTargetEM;
    OK := TRUE;
  END_IF
END_IF
END_IF
```

7.4.5 Initializing the consumers

The functional example contains a total of eight different consumers. Since the default value of the target energy mode for the BasicDevice function blocks is 16#FF ("Operational"), first configure the energy mode "Operational" for all eight consumers. This is done with the "actInit" action. It will run when the program starts until the target energy mode "Operational" has been set for actual/virtual consumers:

Program:

```
IF bfirstSet = FALSE THEN (*Initialisation of the devices is running*)
  actInit();
ELSEIF bfirstSet = TRUE THEN (*Initailisation of the devices ok*)
  (*-----fbBaDevConvHandl: conveyor of the handling module-----*)
  actConvHandl(); (*the action actConvHandl with necessary commands
    for the conveyor of the handling module is executed...*)
```

Realizing EnergyControl

```

(*-----fbBaDevXAxis: linear motor for the X-axis-----*)
actXAxis (); (*the action actConvHandl with necessary commands
for the conveyor of the handling module is executed...*)

(*-----fbBaDevYAxis: linear motor for the Y-axis-----*)
actYAxis (); (*the action actConvHandl with necessary commands
for the conveyor of the handling module is executed...*)

(*-----fbBaDevZAxis: linear motor for the Z-axis-----*)
actZAxis (); (*the action actConvHandl with necessary commands
for the conveyor of the handling module is executed...*)

(*-----fbBaDevLightHandl: lighting of the handling module-----*)
actLightHandl (); (*the action actConvHandl with necessary commands
for the lighting of the handling module is executed...*)

(*-----fbBaDevConvPress: conveyor of the press module-----*)
actConvPress (); (*the action actConvHandl with necessary commands
for the conveyor of the handling module is executed...*)

(*-----fbBaDevPress: press motor-----*)
actPress (); (*the action actConvHandl with necessary commands
for the conveyor of the handling module is executed...*)

(*-----fbBaDevLightPress: lighting of the press module-----*)
actLightPress (); (*the action actConvHandl with necessary commands
for the conveyor of the handling module is executed...*)
END_IF

```

If "bfirstSet" was set to TRUE, initialization is complete and "actInit" is no longer called. From this time on, only the actions for the individual consumers are executed. These then only initiate switching procedures if the target energy modes are changed.

The structure of the "actInit" action is explained in further detail below. The initial switching of the consumers to the operational energy mode is realized for conveyor belts and Press in a simple manner:

Program:

```

(*Conveyor Handling*)
(*output for the conveyor of the handling module:*)
bConvHandl := TRUE; (*first setting only once*)
bfirstSetConvHandl := TRUE;

```

The output for each consumer is set to TRUE. The variable "bFirstSetConvHandl" serves to check the start procedure.

When turning on the axes, make sure that these could also be in parameterization mode:

Program:

```

(*-----X-axis-----*)
udiStatusXAxis:= DV_Axis[1].A_0_0024;
CASE udiStatusXAxis_OF
8: (*Parameterization mode*)
  IF arAxisStatus_gb[1].Admin.ParamLevelAck = FALSE THEN
    arAxisCtrl_gb[1].Admin.ParamLevel := TRUE;
  ELSIF arAxisStatus_gb[1].Admin.ParamLevelAck = TRUE THEN
    timerx (tSwitchtime:=T#1S);
    IF timerX.OK = TRUE THEN
      arAxisCtrl_db[1].Admin.ParamLevel := FALSE;
    END_IF
  0: (*Axis is active*)
    SwitchXAxis (usiTargetEM := 16#FF, uiAxisNumber := 1);
  ELSE (*Simulating, axis is inactive*)
    VirtualXAxis(tSwitchtime:=T#4S); (*This time needs the device for
    setting the target energy mode*)
END_CASE
bfirstSetXAxis := VirtualXAxis.OK OR SwitchXAxis.OK

```

The purpose is to switch the axes to the operational energy mode. If this is in parameterization mode (udiStatusXAxis = 8), the IF instruction first initializes the AxisInterface and parameterization mode is disabled. The axis is then in

Realizing EnergyControl

AB mode (udiStatusXAxis = 0) and the "SwitchAxis" function block is called, to which the operational energy mode is transmitted as the target energy mode along with the axis number (if X-axis, the number is 1). If the axis is inactive (ELSE branch), switching is simulated with the "VirtualDevice" function block. Lastly, use "bfirstSetXAxis" to check whether or not activation was successful. If the "VirtualDevice" or "SwitchAxis" function block reports a successful switching with TRUE, this control variable is set to TRUE accordingly by an OR operation.

The "actInit" action finally checks whether or not the activation of all consumers was successful by using the following assignment:

Program:

```
bfirstSet := bfirstSetConvHandl AND bfirstSetXAxis AND bfirstSetYAxis AND  
            bfirstSetZAxis AND bfirstSetLightHandl AND bfirstSetConvPress AND  
            bfirstSetPress AND bfirstSetLightPress;
```

The AND operation sets the variable **bfirstSet** to TRUE only if the activation procedure was completed for all eight consumers. This ensures that initialization is completed and only the eight actions for the individual consumers are subsequently called.

7.4.6 Switching consumers after initialization

Following initial activation, the eight actions for the consumers are called in cyclical fashion. Switching procedures are only triggered when the target energy mode changes. This is explained using the Handling station conveyor belt:

Program:

```
(*-----fbBaDevConvHandl: conveyor of the handling module-----*)  
(*querying the actual target energy mode with the method mGetControl  
- this method has a return value and two outputs (ChangeEM und  
TargetEM)  
- the outputs are valid only if the function block is in the  
state InOperation, this is indicated by the method  
mGetControl: return value = TRUE  
- if the return value is TRUE:  
-> output ChangeEM: NO_CHANGE or CHANGE_ENERGY_MODE  
-> output TargetEM: hexadecimal code according to the  
configuration*)  
bReturnConvHandl:= IL 4EE.fbBaDevConvHandl.mGetControl(ChangeEM=>  
enChangeEMConvHandl, TargetEM=> usiTargetEMConvHandl);  
IF bReturnConvHandl = TRUE THEN (* the outputs of the  
method mGetControl are valid *)  
  
    IF enChangeEMConvHandl = CHANGE_ENERGY_MODE THEN (*if the target  
energy mode has changed, the output ChangeEM has the value  
CHANGE_ENERGY_MODE. The output ChangeEM remains in the state  
CHANGE_ENERGY_MODE until the presence of the energy mode is  
reported or the command of the Command-FB is cancelled  
-> The energy mode at the output TargetEM = usiTargetEMConvHandl  
must be set*)  
        bConvHandl := (usiTargetEMConvHandl = 16#FF);  
        VirtualConvHandl(tSwitchtime := T#1S); (* This time needs  
the device for setting the target energy mode *)  
        IF VirtualConvHandl.OK=TRUE THEN  
            (*the energy mode has been set, actual energy mode =  
target energy mode*)  
            (*After the setting of the energy mode: feedback of  
the current energy mode to the function block  
fbBaDevConvHandl, in hexadecimal  
code*)  
            IL 4EE.fbBaDevConvHandl.mSetState(CurrentEM :=  
usiTargetEMConvHandl);  
        END_IF  
    END_IF (*NO_CHANGE -> no change of energy mode -> nothing has  
to be done*)  
  
END_IF (*the outputs of the method are not valid*)
```

Realizing EnergyControl

First the current target energy mode is queried via the "mGetControl" method for the BasicDevice block "fbBaDevConvHandl". In addition to the return value, it has the two outputs "ChangeEM" and "TargetEM". They are then only valid if the block is in the "InOperation" state. This is displayed via the method return value. If the return value is TRUE, the "ChangeEM" output is "NO_CHANGE" or "CHANGE_ENERGY_MODE" (enumeration). It indicates whether or not the target energy mode for the BasicDevice function blocks has changed. The "TargetEM" output returns the target energy mode as a coded hexadecimal value.

Now assign the method return value in the program to the "BReturnConvHandl" variables, and also assign the outputs "ChangeEM" and "TargetEM" to variables. If the return value is TRUE and the "ChangeEM" output shows a change to the target energy mode, the `bConvHandl := (usiTargetEMConvHandl = 16#FF)` assignment triggers a switching procedure and the "Virtual-Device" function block is used again to simulate the switching process on the non-existing consumer. The consumer's current energy mode is reported back to the "fbBaDevConvHandl" BasicDevice block with the "mSetState" method. The remaining consumers switch in similar fashion.

7.5 Using commissioning visualization

After declaring and instantiating the blocks, add an empty "OverviewEnergyControl" visualization to the project via **Application ► Insert ► Visualization**. An overview of the blocks used and their nesting will be added here.

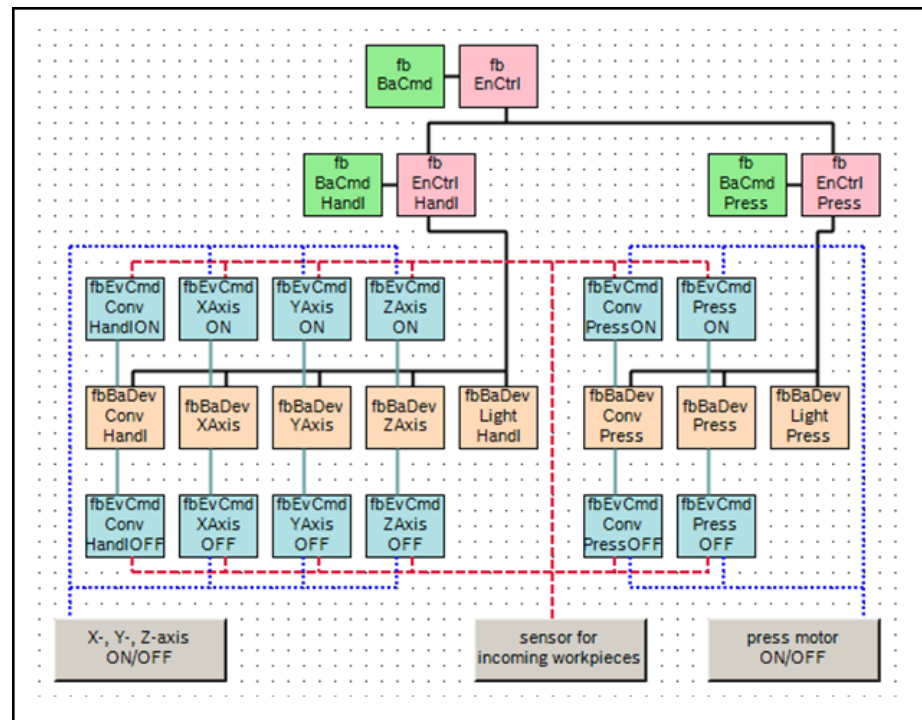


Fig. 7-18: "OverviewEnergyControl" visualization

Clicking on the blocks opens a single view where energy modes and dead times can be specified. This function was added to each block in the overview by selecting it through **Properties ► Entry Configuration ► OneMouse-Click ► Visualization Change** and entering the name of the single view with which a link is desired. The added buttons turn the axes on and off and operate the sensor.

Built-in elements are now available that can be used to operate and display the modes of the blocks:

Realizing EnergyControl

Function block/structure	Visualization
IL_4EEEnergyControlType01	EnergyControlType01
IL_4EEBasicCommandType01	BasicCommandType01
IL_4EEEventCommandType01	EventCommandType01
IL_4EEEventCommandType02	EventCommandType02
IL_4EEBasicDeviceType01	BasicDeviceType01
Energy mode Array Indexes	Array Indexes

Tab. 7-1: Available visualization elements for the function blocks

These visualizations were first added to the project under the menu item **Application** ► **Data Transfer**. This allows a frame to be inserted in the single view of each block via **Tools** ► **Frame**. The desired visualization was selected with the Frame option. This can then be linked to the corresponding instance that is to be displayed with the added visualization element via **Properties** ► **Referenced Visualization**. The necessary elements can be added to each visualization in this manner.

The "Enable" button in each block changes the block's processing enable. The user interface for each block also provides other control options.

EnergyControlType01: The "CurrentEM" entry field and the button with the "mSetState" method can return the current energy mode. This requires the value of the "StatusOverride" element to be TRUE in the configuration.

IL_4EEEnergyControlType01		
Handling module		
FB Interface	Status	
Enable ☒	ChangeEM	NO_CHANGE
InOperation ☒	TargetEM	FF
Error ☐	CurrentEM	FF
Methods		
	StatusOverride	FALSE
mSetState	CurrentEM	0

Fig. 7-19: Visualization of the EnergyControl function block

BasicCommandType01: The button with the "mSetDirectCommand" method and the "TargetEM" entry field can directly specify an energy mode. The "mSetPauseCommand" button and the "PauseTime" entry field can set dead times. The "WakeUpMode" entry field can enable/disable this function. The "mResetCommand" method can rescind the current command.

Realizing EnergyControl

IL_4EEBasicCommandType01		
Whole machine		
FB Interface		Status
Enable	☒	State
InOperation	☒	TargetEM
Error	☐	
Methods		
mSetDirectCommand	TargetEM	0
mSetPauseCommand	PauseTime	T#0ms
	WakeUpMode	FALSE
mResetCommand		

Fig. 7-20: Visualization of the BasicCommand function block

EventCommandType01 and EventCommandType02: "mSetEventCommand" can specify the target energy mode entered during configuration. "mResetCommand" can rescind this command.

IL_4EEEventCommandType01		
Conv Handl OFF		
FB Interface		Status
Enable	☒	Status
InOperation	☒	TargetEM
Error	☐	RemainingTime
Methods		
mSetEventCommand		
		mResetCommand

Fig. 7-21: Visualization of the EventCommand function block

IL_4EEBasicDeviceType01: The button with the "mSetState" method and the "CurrentEM" entry field can return the consumer's current energy mode.

IL_4EEBasicDeviceType01		
Conveyor Handl		
FB Interface		Status
Enable	☒	ChangeEM
InOperation	☒	TargetEM
Error	☐	CurrentEM
Methods		
mSetState	CurrentEM	0

Fig. 7-22: Visualization of the BasicDevice function block

7.6 Visualizing "SmartAutomation"

This visualization shows the operating mode of the entire machine and reacts to changes in the EnergyManagement. Control variables, trigonometric functions, the Module Operator and other tools are used to move the individual elements. The "actVisu" action and the "SmartAutomation" visualization contain the commands needed for this.

Realizing EnergyControl

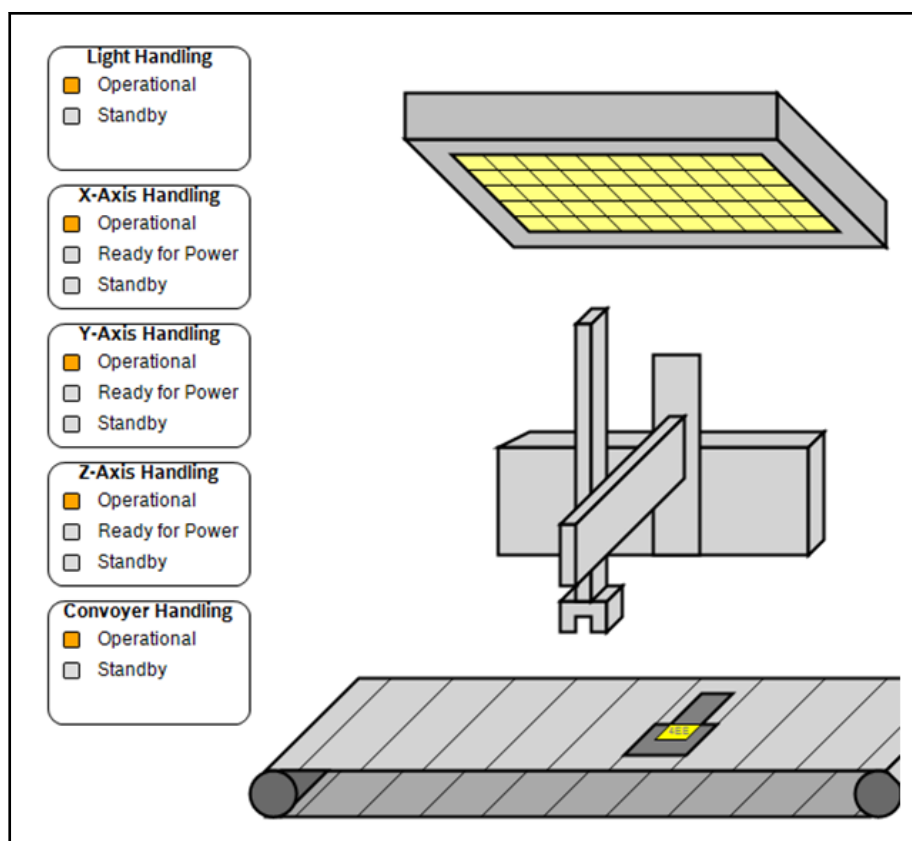


Fig. 7-23: Visualization of the Handling module

Realizing EnergyControl

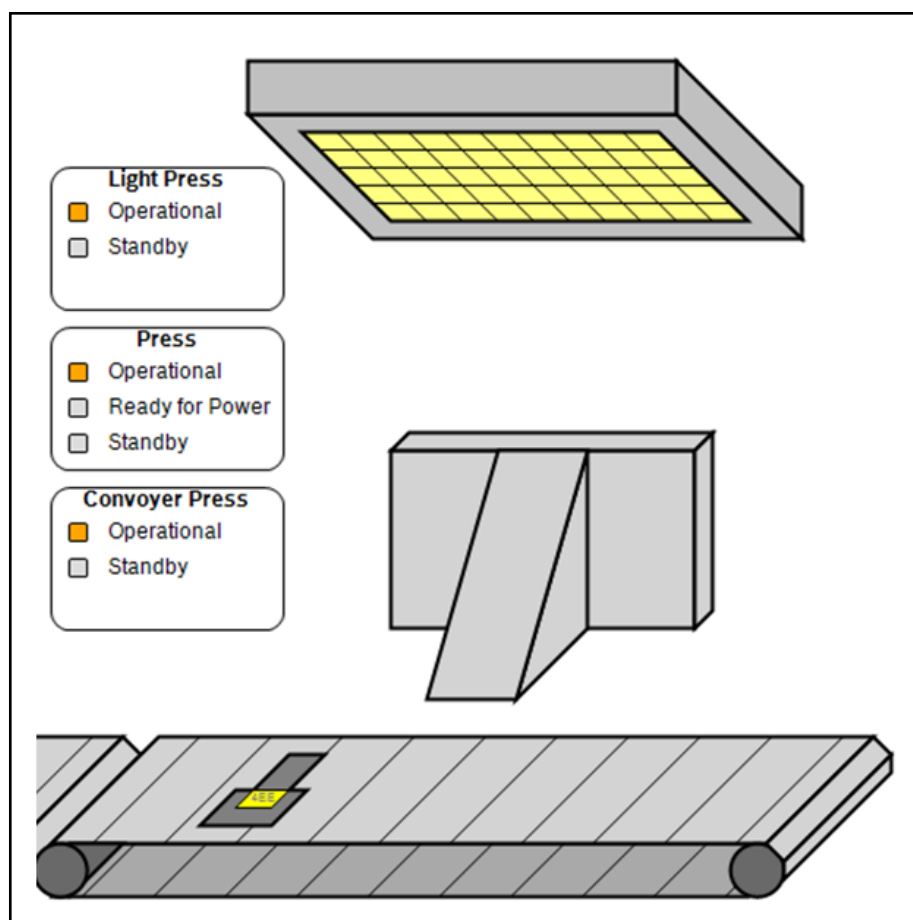


Fig. 7-24: Visualization of the Press module

8 HMI Controls

8.1 IndraWorks Operation Desktop

8.1.1 EnergyMonitor

Overview

Purpose of EnergyMonitor

With the EnergyMonitor, the energy consumption of a machine and individual function modules (electrical drives and ancillary components, such as hydraulic pumps, coolant/lubricant pumps, etc.) can be tallied and monitored over defined time periods.

The EnergyMonitor is part of the MTX interface.

EnergyMonitor properties

The EnergyMonitor shows the energy consumption:

- For the last 10 measurements in the form of a bar chart
- Of individual drives
- Of individual ancillary components

Components

The EnergyMonitor consists of the following sections:

- Header
- Main screen
- Function key bars

Commissioning the EnergyMonitor

Requirements

The EnergyMonitor obtains the data it displays from the PLC function block "MT_4EEGuiMonitor". The relevant function block has to be integrated and assigned in the PLC program for this purpose see (PLC Programming Guide).

Working with the EnergyMonitor

Starting the EnergyMonitor

The EnergyMonitor is started from Production Data by pressing <F2>. The EnergyMonitor can also be mapped to F keys or M keys on user screens.

Total energy consumption

After opening the EnergyMonitor, the last 10 energy measurements are shown in a bar chart, whereby the latest measurement is always shown on the right side. Each individual measurement shows the consumption percentage of the Sercos drives, as well as the ancillary components entered. If necessary, pressing <F2> can optionally display the duration of measurement. This is very helpful for monitoring in Production/Part Manufacturing, in particular. Functions for running/disabling energy and time measurements can optionally be mapped to the M keys.

HMI Controls

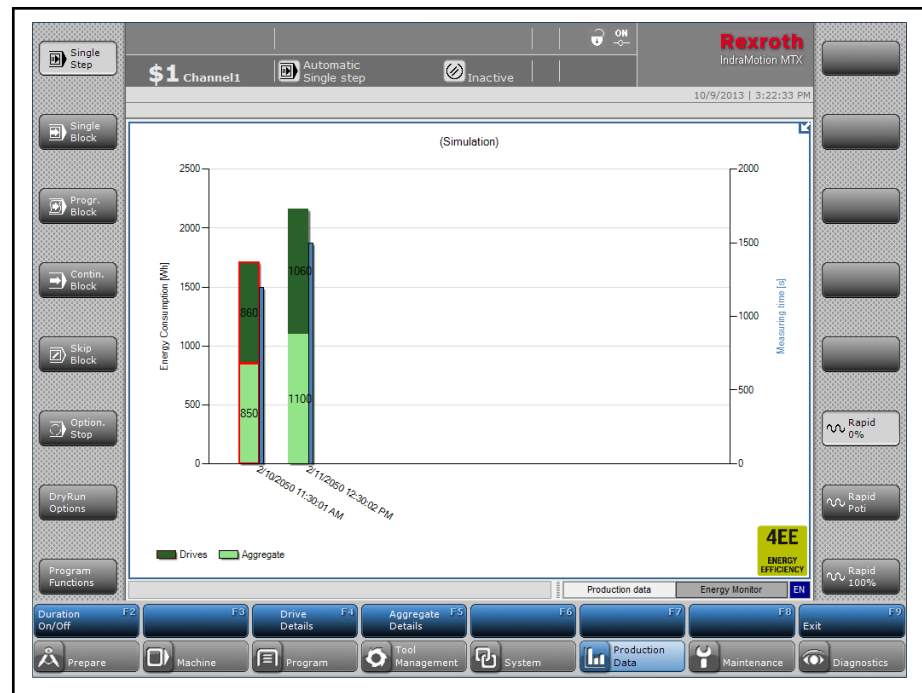


Fig. 8-1: Total energy consumption

The individual energy measurements can be selected from the menu with the arrow keys. If necessary, the selected measurement can be analyzed in greater detail. Pressing <F4> shows the energy consumption of the individual drives, and <F5> the energy consumption of the individual ancillary components in detail.

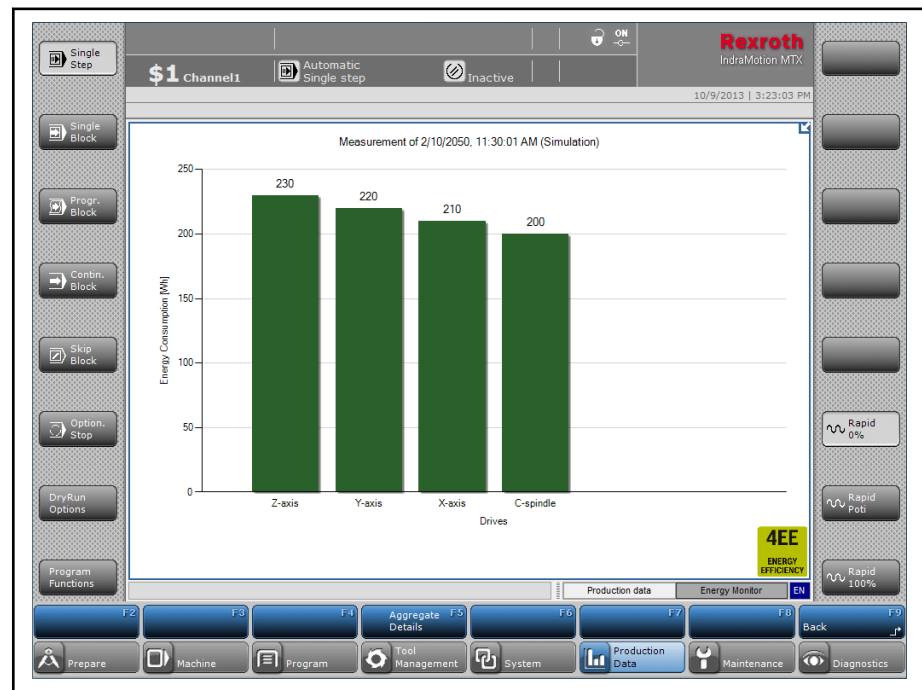


Fig. 8-2: Energy consumption of drives

Energy consumption of drives

Pressing <F4> opens the menu for displaying the energy consumption of the individual Sercos drives. The energy consumption bars for the individual drives can be accessed from the menu with the arrow keys.

Energy consumption of ancillary components

Pressing <F5> opens the menu for displaying the energy consumption of the individual ancillary components. The energy consumption bars for the individual ancillary components can be accessed from the menu with the arrow keys.

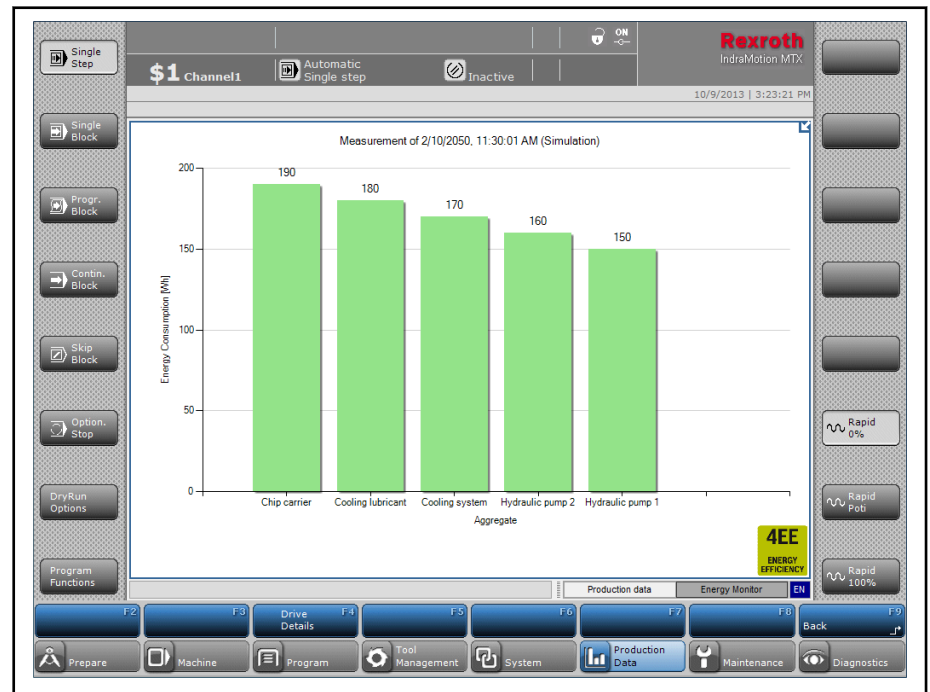


Fig. 8-3: Energy consumption of ancillary components

8.1.2 PowerMonitor Overview

Purpose of the PowerMonitor

With the PowerMonitor, the power consumption (effective power) of a machine and individual function modules (electrical drives and ancillary components, such as hydraulic pumps, coolant/lubricant pumps, etc.) can be observed and monitored.

The PowerMonitor is part of the MTX interface.

PowerMonitor properties

The PowerMonitor shows:

- The overall power of the machine in either a chronologically variable bar chart or in a timeline
- The output power of the individual drives
- The output power of the individual ancillary components

Components

The PowerMonitor consists of the following sections:

- NC main screen subwindow
- Function key bars

HMI Controls

Commissioning the PowerMonitor

Requirements

The PowerMonitor obtains the data it displays from the PLC function block "MT_4EEGuiMonitor". The relevant function block has to be integrated and assigned in the PLC program for this purpose see (PLC Programming Guide).

The PowerMonitor also has to be configured on the default main screen. To do this, first open the PowerMonitor from the default main screen via the second F key level after selecting the program display window with <F7> and <F3>, display the configuration menu with <F8>, and then edit the configuration menu according to the following subchapters.

Configuring the PowerMonitor - General information

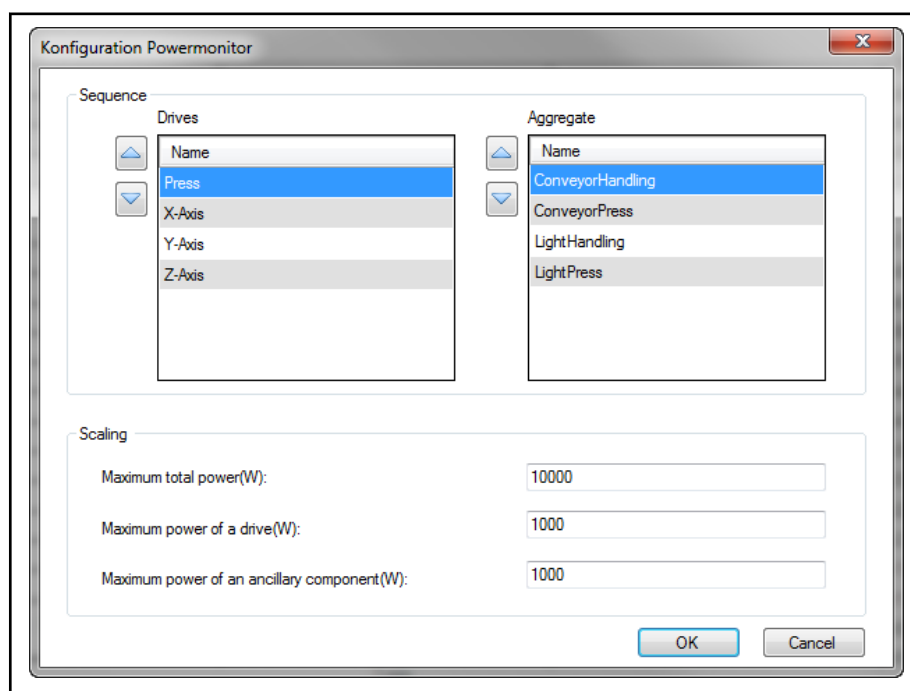


Fig. 8-4: Configuring the PowerMonitor - General information

Enter the maximum total drive power, the maximum single drive power, and the maximum output power for each individual ancillary component under "General". The display order of the drives and ancillary components can also be specified for the various power displays by using the arrow keys here as desired.

Working with the PowerMonitor

Starting the PowerMonitor

The PowerMonitor can be displayed as needed from the default main screen via the second F key level with <F7> and <F3>.

The PowerMonitor can also be mapped to F keys or M keys on user screens.

Overall power



Fig. 8-5: Overall power (as timeline)

The overall power of the machine can be displayed in either a chronologically variable bar chart or in a timeline. Both display types show both the consumption of the drives and the consumption of the ancillary components. Pressing <F2> switches between the two displays.

Output power of the drives

Pressing <F4> opens the power display menu for the individual Sercos drives. The power bars for the individual drives can be accessed from the menu with the arrow keys.

HMI Controls

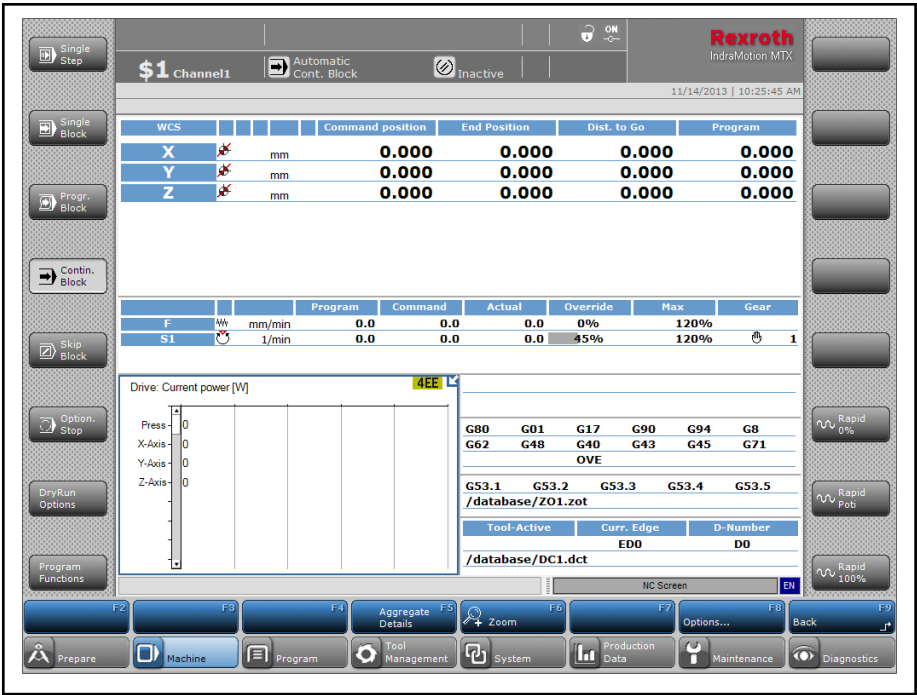


Fig. 8-6: Output power of the drives

Output power of the ancillary components

Pressing <F5> opens the power display menu for the individual ancillary components. The power bars for the individual ancillary components can be accessed from the menu with the arrow keys.



Fig. 8-7: Output power of the ancillary components

Zoom

If necessary, pressing <F6> enlarges each display. This is especially recommended for displaying the individual output power stats for the drives as well as for the individual ancillary components.

Pressing <F6> again restores the original display size.

8.1.3 EnergyManagement

Overview

Purpose of EnergyManagement

With the EnergyManagement, control and status data can be followed at the machine and component level. It can also monitor the statuses of each event and energy saving functions.

The EnergyManagement is part of the MTX interface.

EnergyManagement properties

The EnergyManagement has the following screens:

- Machine information
- Component information
- Events
- Energy saving functions (for individual components incl. those in the machine)
- Dead time/energy mode specification

Components

The EnergyManagement screens can be found under **Production Data ► EnergyManagement**.

Working with the EnergyManagement (dead time or energy mode)

Requirements

The dead time/energy mode can be specified under Production Data. To do this, open the "Dead Time or Energy Mode" window with <F7>.

Specifying the dead time or the energy mode

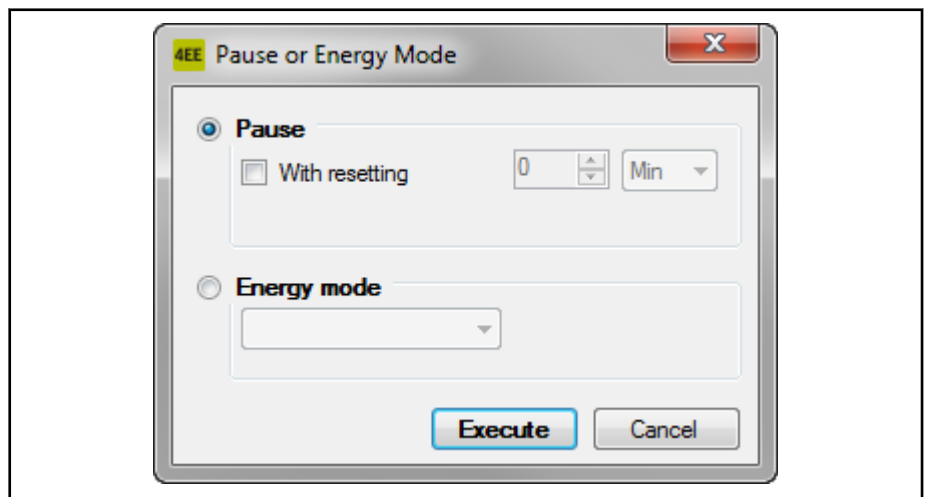


Fig. 8-8: Configuring "Dead Time or Energy Mode" window

Either a dead time or an energy mode can be specified in the window. If a break is desired, the checkbox can be used to select dead time with or without restart. If dead time with restart is selected, the system automatically switches to energy mode "Operational" once the dead time has elapsed.

HMI Controls

Working with the EnergyManagement

Machine information

Pressing <F2> opens the menu for displaying the machine information.

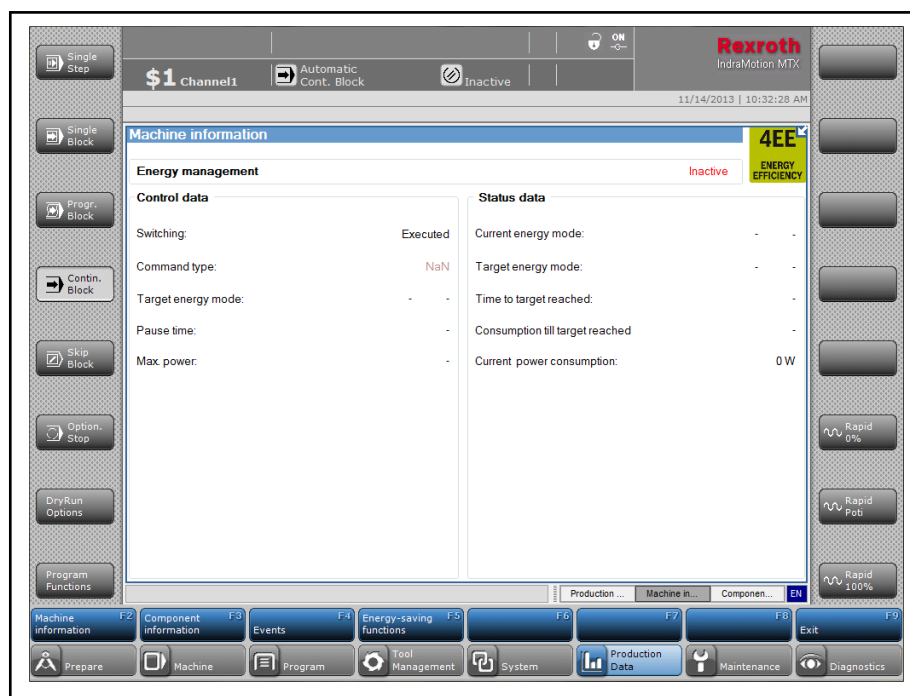


Fig. 8-9: Machine information

The menu shows both the control data and status data for the machine.

The control data contains the following items:

- "Switching" shows whether or not energy mode switching is active
- "Command type" indicates the type of specification (direct specification of a desired energy mode or dead time with or without restart)
- "Target energy mode" specifies the energy mode to be initiated, provided the command type is a direct energy mode specification
- "Dead time" shows the duration of dead time. It is used in the Energy-Management to calculate the target energy mode

The EnergyManagement displays the following for the machine under the status data:

- "Current energy mode" shows the active energy mode of the machine
- "Target energy mode" specifies the energy mode that the EnergyManagement has calculated. It is possible that this is not the same as the target energy mode specified by the user (on the left side). In these instances, another, higher-priority specification is typically pending (e.g., from an MES system)
- "Time to target mode" shows the time left before switching
- "Consumption to target mode" shows the energy consumption left before switching
- "Current power consumption" indicates the average output power of the active energy mode

Component information

Pressing <F3> opens the menu for displaying the component information.

HMI Controls



Fig. 8-10: *Component information*

The menu shows whether or not the mode for each component has to be switched, as well as the assigned target energy mode incl. current energy mode.

Events

Pressing <F4> opens the menu for displaying events.



Fig. 8-11: *Events*

The menu shows whether or not each event is active, as well as the assigned target energy mode incl. delay time and remaining time until the event becomes active.

HMI Controls

Energy saving functions

Pressing <F5> opens the menu for displaying the energy saving functions.



Fig. 8-12: Energy saving functions

The menu shows each energy saving function, the components to which they belong, and the partial load operation.

8.2 WinStudio

8.2.1 Overview

The following .Net controls are available for the EnergyManagement:

Range	Name of controls	Description
EnergyMonitoring	EnergyMonitorACI-NETCtrl	Represents the measured values of the energy measurement
	PowerMonitorACI-NETCtrl	Representation of the current performance data
EnergyControl	EMngMachineControlACI-NETCtrl	Status of the machine
	EMngDeviceControlACI-NETCtrl	Status of the individual consumers
	EventControlEMACI-NETCtrl	Status of the EventCommand function blocks
	EMngEnergyFunction-sACI-NETCtrl	Status of the EnergyFunctions

Tab. 8-1: .NetControls overview

The Controls can be integrated in WinStudio. The Controls communicate via the OPC.IwSCP-Server. The server has to be configured. The following structures have to be instantiated on the control with the namespace for the controls and have to be available in the symbol configuration:

EnergyMonitoring

- st4EEMonConfig, of type IL_4EE_GUI_MON_CONFIG
- st4EEMonResult, of type IL_4EE_GUI_MON_RESULT_MIN

Namespace: IL_4EE_Energy_Power_Monitoring

EnergyControl

- IL_4EE_HMI, of type IL_4EE_HMI_DATA_TYPE01

Namespace: Application.IL_4EE_ENERGY_CONTROL

This documentation describes the integration of .NetControls into WinStudio. It is also possible to integrate .NetControls into other visualization environments. To communicate via the OPC.lwSCP server, install the IndraWorks package "Communication - IndraLogic, IndraMotion MLC" first.

8.2.2 Inserting .NET controls in WinStudio

The .Net controls are contained in the HMI.Energy.StandAlone.dll assembly in the ...\\IndraWorks\\WinStudio\\bin directory in the IndraWorks installation.

Execute the following steps to configure a control in a WinStudio image:

- Create WinStudio application as HMI device
- Create new WinStudio image
- Select „**.NET Control**“ under WinStudio - Screen Tools

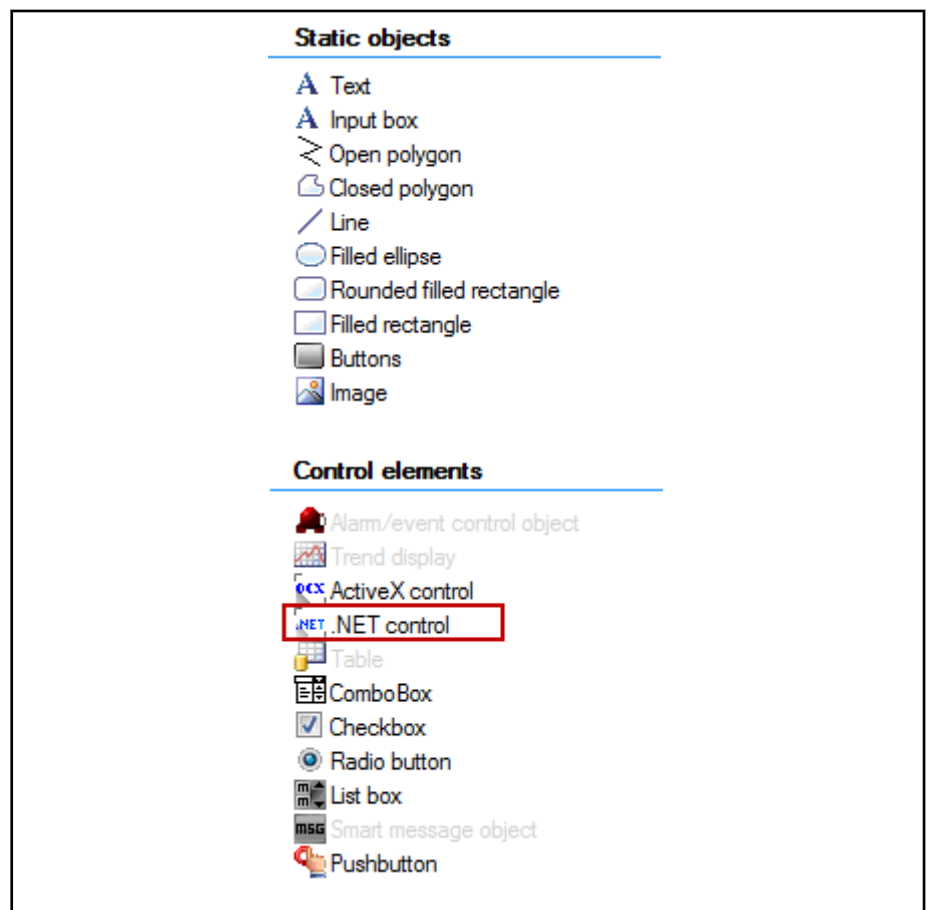


Fig. 8-13: .NET Control in the menu

The .NET framework component dialog is opened.

HMI Controls

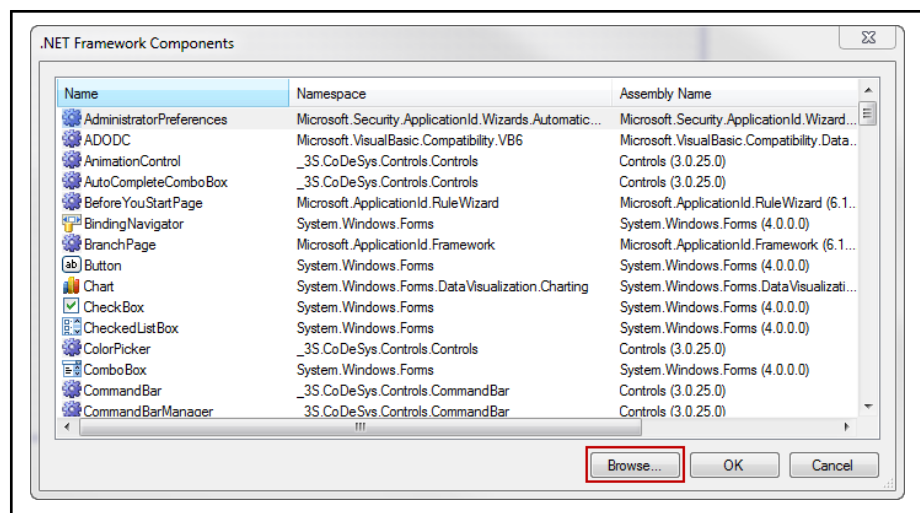


Fig. 8-14: .NET Framework components

Show the "Open" dialog by clicking on **Browse...**. Select the HMI.Energy.StandAlone.dll file in the ...\\IndraWorks\\WinStudio\\bin directory.

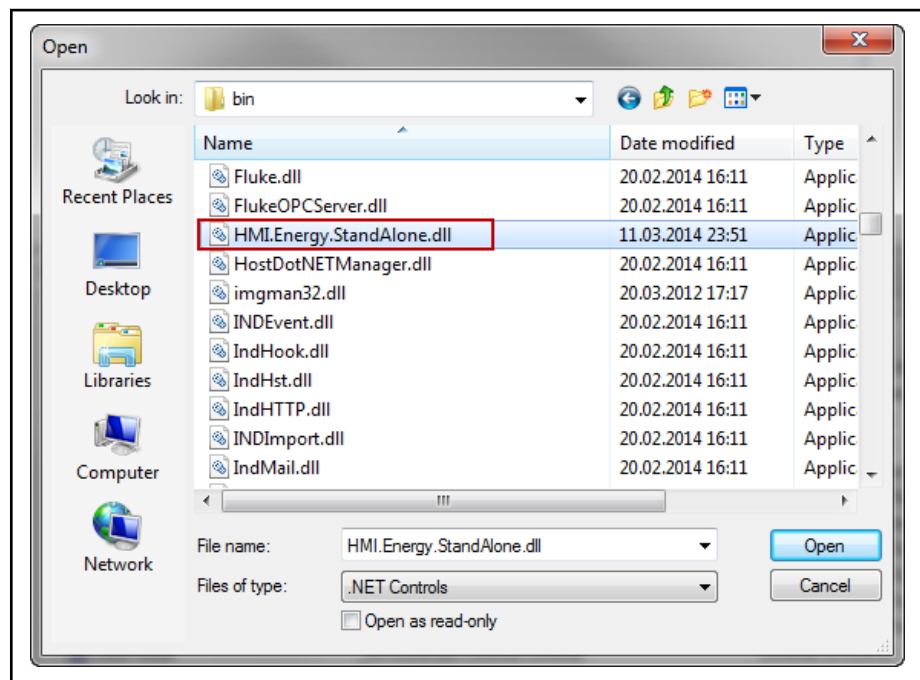


Fig. 8-15: Open file dialog

Six controls with identical namespace are now listed in the ".NET Framework components" dialog.

HMI Controls

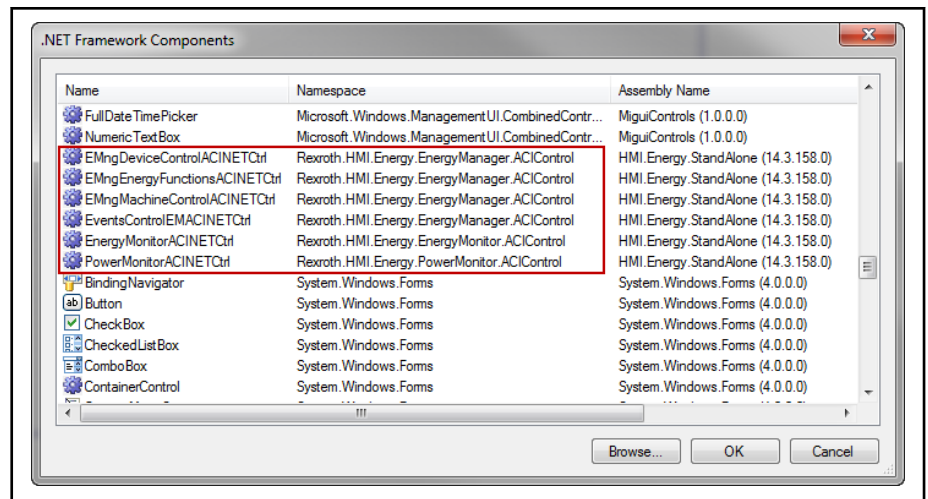


Fig. 8-16: .NET Framework components dialog with six controls

By selecting the control and confirming with **OK**, the controls can now be inserted and scaled in the WinStudio image.

8.2.3 Control interfaces and operation

General information

All 4EE controls are equipped with interfaces for activation and error diagnostics as well as with specific interfaces to select different views. The interfaces can be connected via WinStudio image scrips (e.g. Screen_OnOpen), scripts of pushbuttons in WinStudio or by connecting .NET controls to variables under **Properties ► Elements**.

Before switching the device on, the name of the control on which the Energy-Management is running has to be transmitted to all .NET controls via the Device interface. This name has to correspond to the name in the OPC.IwSCP server configuration. Subsequently, .NET control can be activated via the Active interface.

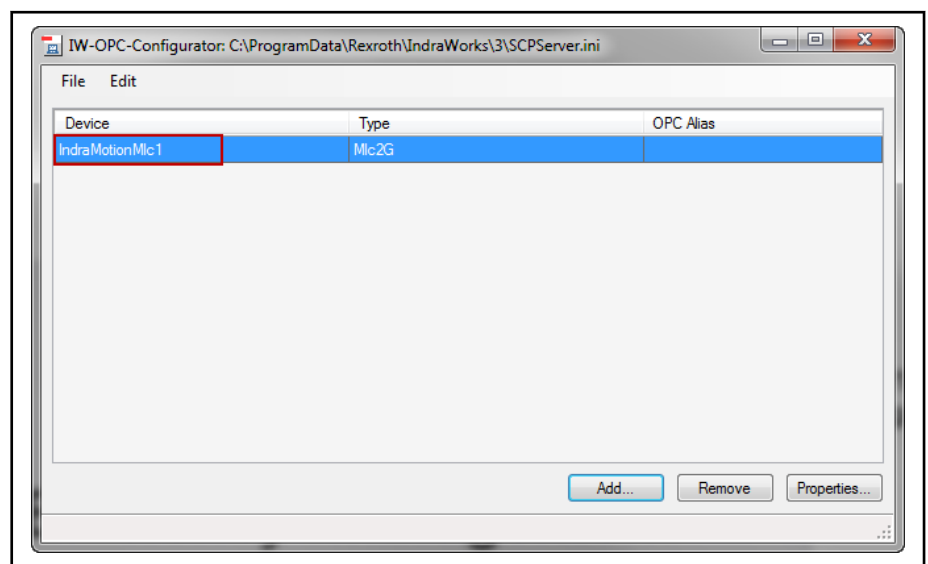


Fig. 8-17: Device in the SCP configuration
Activating the .NET controls in WinStudio

```
$XSet("EMngDeviceControlACINETCtrl1", "Device", "IndraMotionMlc1")
$XSet("EMngDeviceControlACINETCtrl1", "Active", "True")
```

HMI Controls

Errors of all controls are output via the HasError (BOOL) and LastError (STRING, error text) properties. To query these properties, the LastErrorChanged event is available. For this event, a script can be created to which the variable properties are copied.

Error query of .NET controls in WinStudio

```
bError = $XGet("EMngDeviceControlACINETCtrl1", "HasError")
strError = $XGet("EMngDeviceControlACINETCtrl1", "LastError")
```

EnergyMonitor

The .NET control EnergyMonitor can be selected as EnergyMonitorACINETCtrl in the ".NET framework components" dialog and can be inserted in the WinStudio image. The results of the last 10 energy measurements are displayed in this control. The measurements are started and stopped at the IL_4EEMonHMIDataCollectorType01 function block.

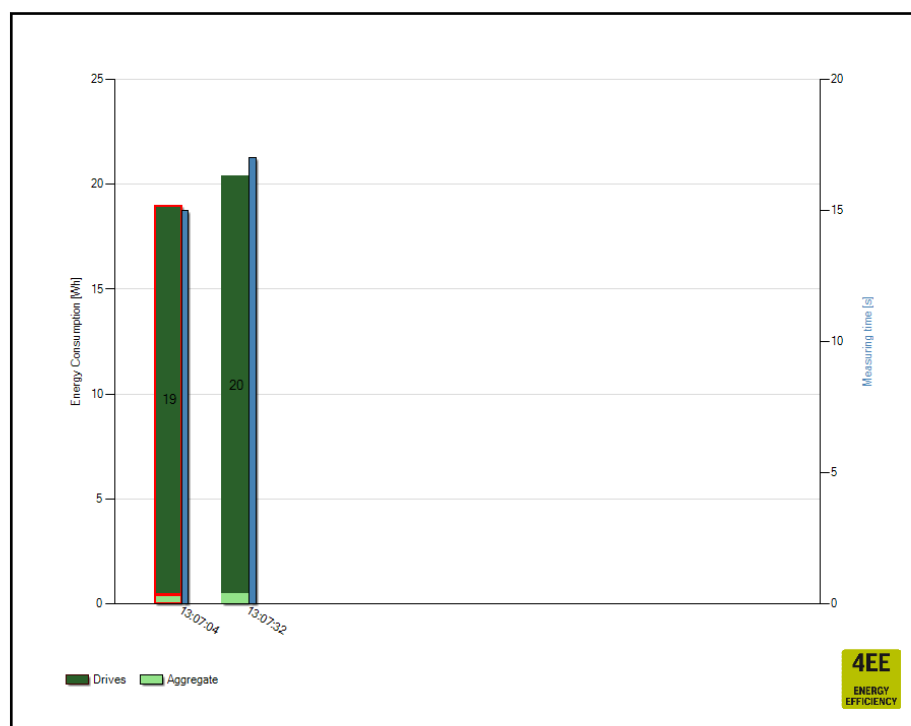


Fig. 8-18: Overview of the .NET control EnergyMonitor

The measurement duration display can be switched on and off via the ShowDuration interface. The graphs of individual measurements can be selected via touch operation. The measured values of the individual drives and aggregates are displayed via the DisplayMode interface.

Interface	Data type	Description
Device	STRING	The name of the control configured in the OPC.lwSCP server
Active	BOOL	Starts the control with TRUE
DisplayMode	ENUM	Control view: • Overview • Drives • Components (Aggregate)

HMI Controls

ShowDuration	BOOL	Displays the duration of the measurement if the value is set to TRUE
HasError	BOOL	Has the value TRUE if an error occurred
LastError	STRING	Error text of the last error
LastErrorChanged	EVENT	Is triggered in case of error or if the error text changes

Tab. 8-2: Interface of the EnergyMonitor control

PowerMonitor

The .NET control PowerMonitor can be selected as PowerMonitorACINETCtrl in the ".NET Framework components" dialog and can be inserted in the WinStudio image. The current energy of the consumers is displayed in this control.

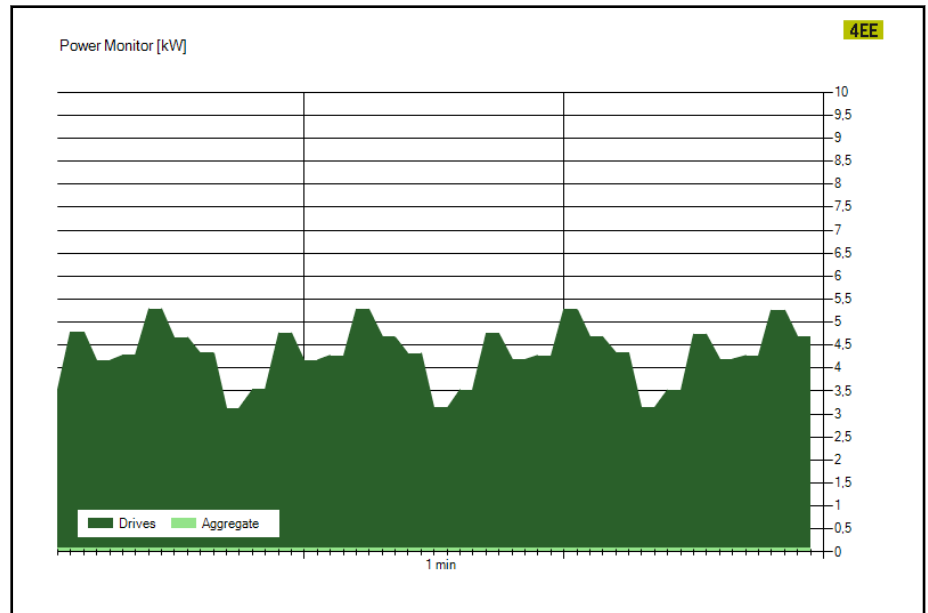


Fig. 8-19: Overview of the .NET control PowerMonitor

In the PowerMonitor overview, toggling between the display of the chronological sequence of the last 2 minutes and the display of current values is possible via the GraphMode interface. The current values of individual drives or aggregates is displayed via the DisplayMode interface.

Interface	Data type	Description
Device	STRING	The name of the control configured in the OPC.lwSCP server
Active	BOOL	Starts the control with TRUE
DisplayMode	ENUM	Control view: • Overview • Drives • Components (Aggregate)

HMI Controls

GraphMode	ENUM	Display of energy in the overview • Default (chronological sequence) • Overview (only current values)
HasError	BOOL	Has the value TRUE if an error occurred
LastError	STRING	Error text of the last error
LastErrorChanged	EVENT	Is triggered in case of error or if the error text changes

Tab. 8-3: Interface of the EnergyMonitor control

Machine

The .NET control machine can be selected as EMngMachineControlACI-NETCtrl in the ".NET Framework components" dialog and can be inserted in the WinStudio image. The state of the EnergyControl function block in which the HMIDataCollector function block is registered in is displayed in this control.

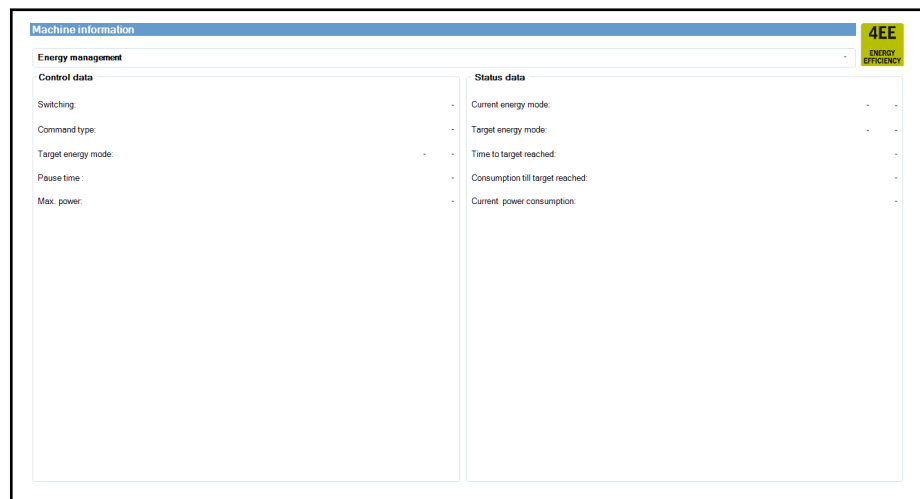


Fig. 8-20: .NET control machine

The MachineControlDialog of the energy management can be called via the ShowMachineControlDialog(STRING Device) method.

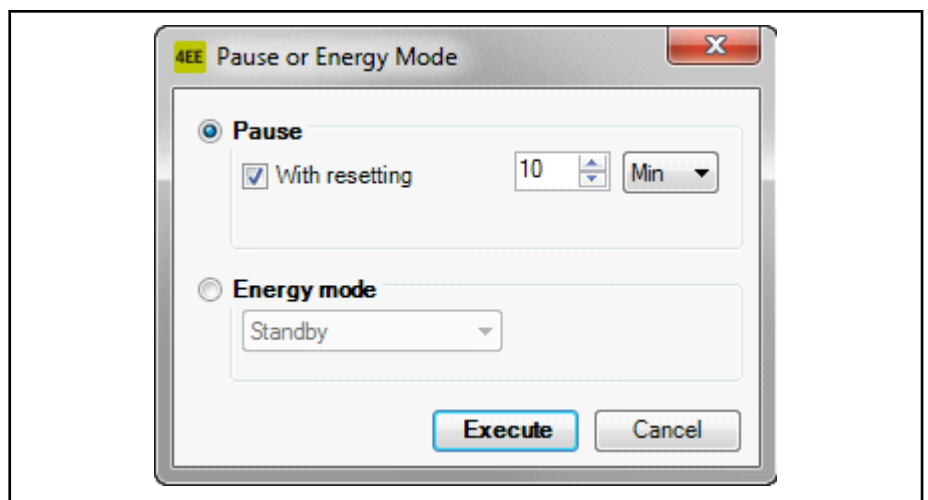


Fig. 8-21: MachineControlDialog
 Calls the control dialog in WinStudio

```
$XRun("EMngMachineControlACINETCtrl1", "ShowMachineControlDialog")
```

The following interfaces are available to operate the control:

Interface	Data type	Description
Device	STRING	The name of the control configured in the OPC.lwSCP server
Active	BOOL	Starts the control with TRUE
HasError	BOOL	Has the value TRUE if an error occurred
LastError	STRING	Error text of the last error
LastErrorChanged	EVENT	Is triggered in case of error or if the error text changes

Tab. 8-4: Interface of the EnergyMonitor control

Devices

The .NET control machine can be selected as EMngDeviceControlACINETCtrl in the ".NET Framework components" dialog and can be inserted in the WinStudio image. The status of all consumers is displayed in the control.

HMI Controls

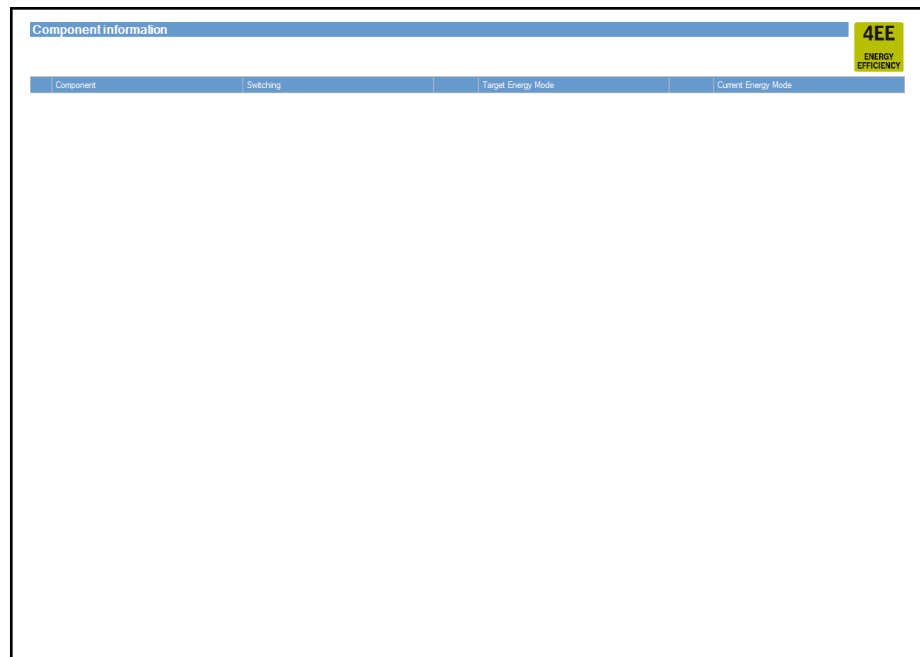


Fig. 8-22: .NET_Control devices

The following interfaces are available to operate the control:

Interface	Data type	Description
Device	STRING	The name of the control configured in the OPC.lwSCP server
Active	BOOL	Starts the control with TRUE
HasError	BOOL	Has the value TRUE if an error occurred
LastError	STRING	Error text of the last error
LastErrorChanged	EVENT	Is triggered in case of error or if the error text changes

Tab. 8-5: Interface of the EnergyMonitor control

Events

The .NET control events can be selected as EventsControlACINETCtrl in the ".NET Framework components" dialog and can be inserted in the WinStudio image. The status of all EventCommand function blocks is displayed in the control.

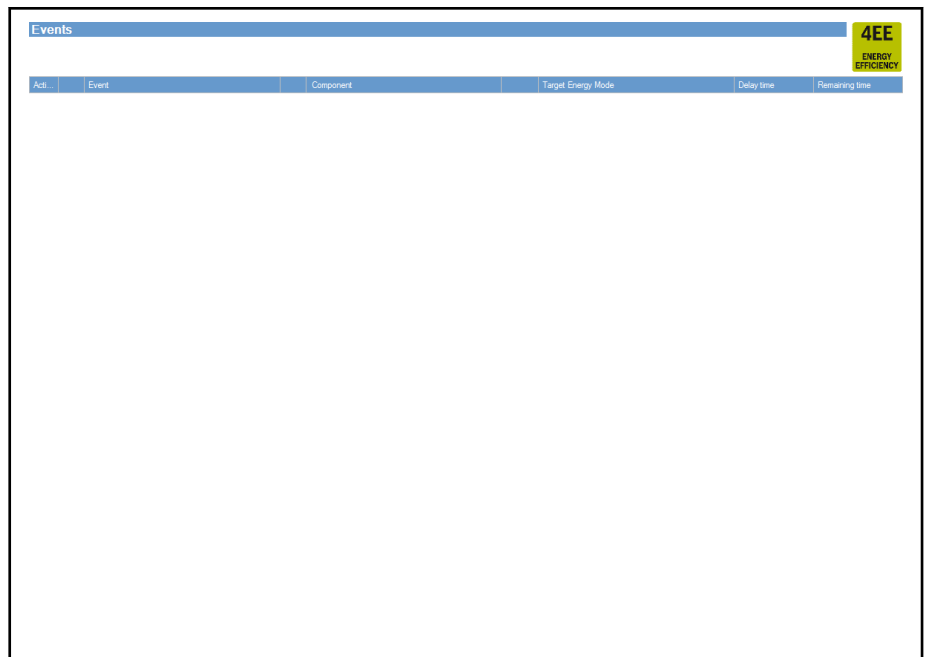


Fig. 8-23: .NET control events

The following interfaces are available to operate the control:

Interface	Data type	Description
Device	STRING	The name of the control configured in the OPC.lwSCP server
Active	BOOL	Starts the control with TRUE
HasError	BOOL	Has the value TRUE if an error occurred
LastError	STRING	Error text of the last error
LastErrorChanged	EVENT	Is triggered in case of error or if the error text changes

Tab. 8-6: Interface of the EnergyMonitor control

EnergyFunctions

The .NET control EnergyFunctions can be selected as EMngEnergyFunction-sACINETCtrl in the ".NET Framework components" dialog and can be inserted in the WinStudio image. The status of all EnergyFunction function blocks is displayed in the control.

HMI Controls

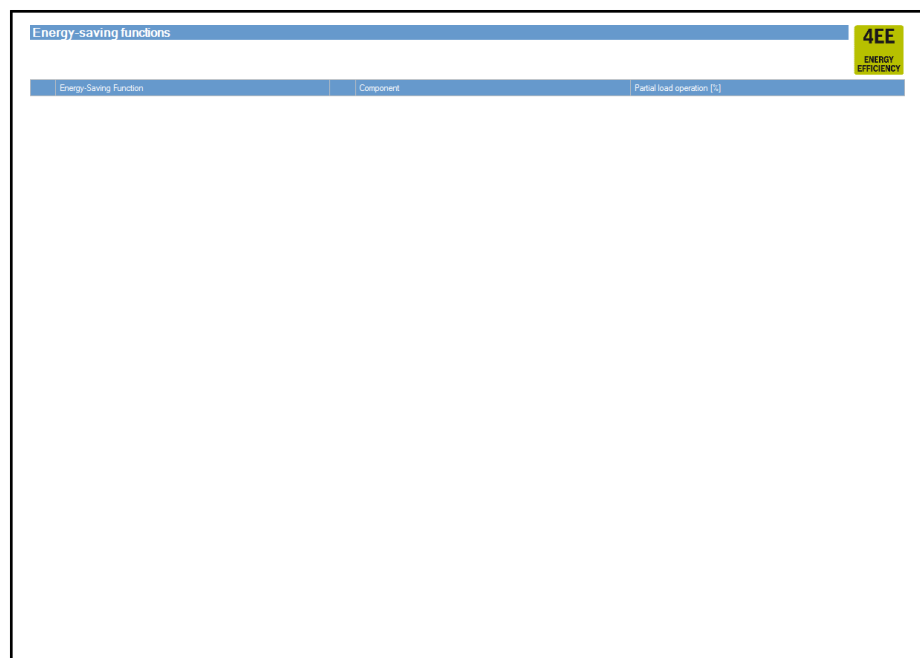


Fig. 8-24: .NET control EnergyFunctions

The following interfaces are available to operate the control:

Interface	Data type	Description
Device	STRING	The name of the control configured in the OPC.lwSCP server
Active	BOOL	Starts the control with TRUE
HasError	BOOL	Has the value TRUE if an error occurred
LastError	STRING	Error text of the last error
LastErrorChanged	EVENT	Is triggered in case of error or if the error text changes

Tab. 8-7: Interface of the EnergyMonitor control

8.3 Web-based visualization

8.3.1 Overview

Introduction To display and analyze the data and the states of the energy management, the web-based visualization is available. The web-based visualization is based on the WebConnector, serving as web server and to communicate with the control. A browser or a browser control displays the data.

Prerequisites An IPC or PC is required on which the WebConnector and an HTML5-compatible browser are installed (e.g. Internet Explorer 11, Firefox 41, Chrome 45) apart from the energy management function blocks to use the web-based visualization.

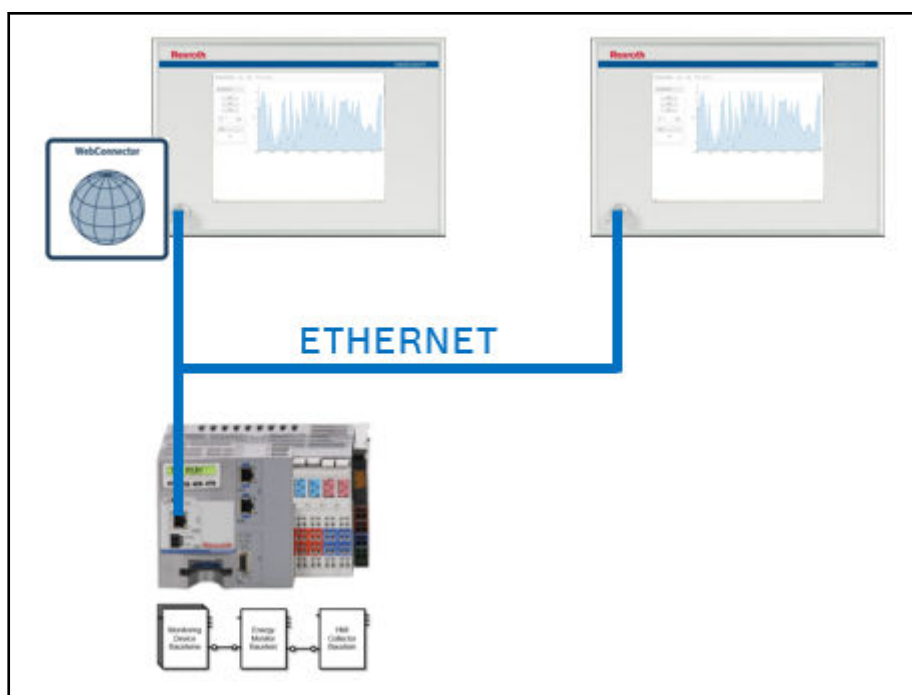


Fig. 8-25: Web visualization architecture

8.3.2 EnergyMonitoring

Overview

Functionality of the web-based visualization

The web-based visualization is used to display and analyze energy and performance measurement data. This data is determined and provided via the monitoring function blocks of the RIL_4EEEnergyManagement library. The web-based visualization reads out the data from the PLC via the WebConnector.

The data of the last ten energy measurements are displayed in a bar diagram. To analyze the data, the ratio of individual consumers or consumer groups of the energy consumption are represented in pie or bar diagrams.

The performance data is represented in a line diagram. Data with an adjustable cycle time is read out of the PLC and buffered in the visualization. To analyze the performance data, the update is paused and the displayed consumers are filtered.

Commissioning the web-based visualization

In the first step, the function blocks of EnergyMonitoring have to be used on the PLC. This means, an instance of a device function block per consumer at which the group associations are configured. The device function blocks are registered at the EnergyMonitor function block that collects the data. Use the IL_4EEMonHMIDataCollectorType02 (without language switching) or IL_4EEMonHMIDataCollectorType03 (with language switching) (see [chapter 9.2.1 "Overview" on page 111](#)) to prepare the interface for the web-based visualization.

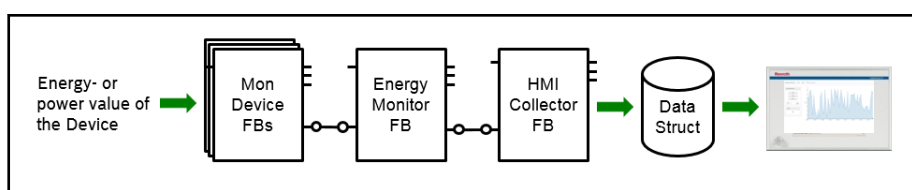


Fig. 8-26: Structure of the EnergyMonitoring on the PLC

HMI Controls

The interface structure IL_4EE_MON_HMI_TYPE02 or IL_4EE_MON_HMI_TYPE03 has to be connected to the HMIDataCollector function block. The default namespace of the interface structure is IL_4EE_ENERGY_MONITORING, the default instance name is st4EEMon-Data.

In the second step, the WebConnector has to be copied and unzipped on the PC or the IPC. Subsequently, the archive "...Add ons\4EE\EnergyMonitoring.zip" has to be unzipped from the IndraWorks installation CD in the WEB-ROOT directory of the WebConnector.

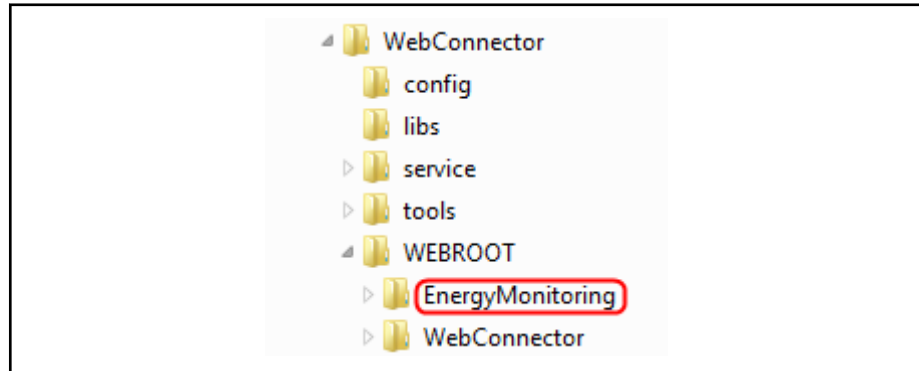
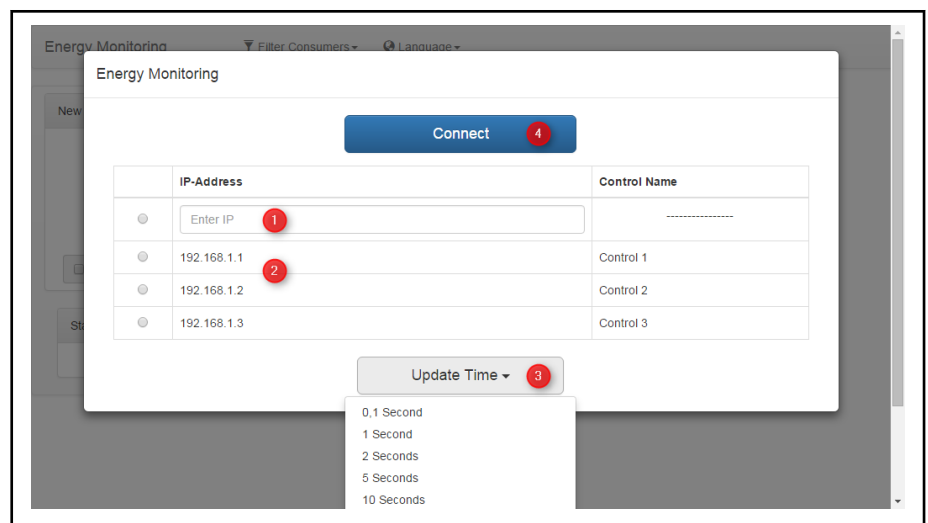


Fig. 8-27: Visualization storage in the WebConnector

In the third step, the Web Connector is started by executing the "...WebConnector\start.bat" file. The web-based visualization can now be called in the browser window via the "http://localhost:8085/EnergyMonitoring/Main-Page.html" address. Use the IP address of the PC the WebConnector is installed on instead of "localhost" to call the visualization from another PC.

Subsequently, the connection window of the web-based visualization opens. Enter the IP address of the control in this window. For manual input, the visualization always uses the default namespace and the default instance name of the interface structure. Optionally, an IP address can be selected. However, this structure has to be edited first in the "...WebConnector\WEBROOT\EnergyMonitoring\json\installedControllers.json" file. Other namespace and instance names can be assigned here.

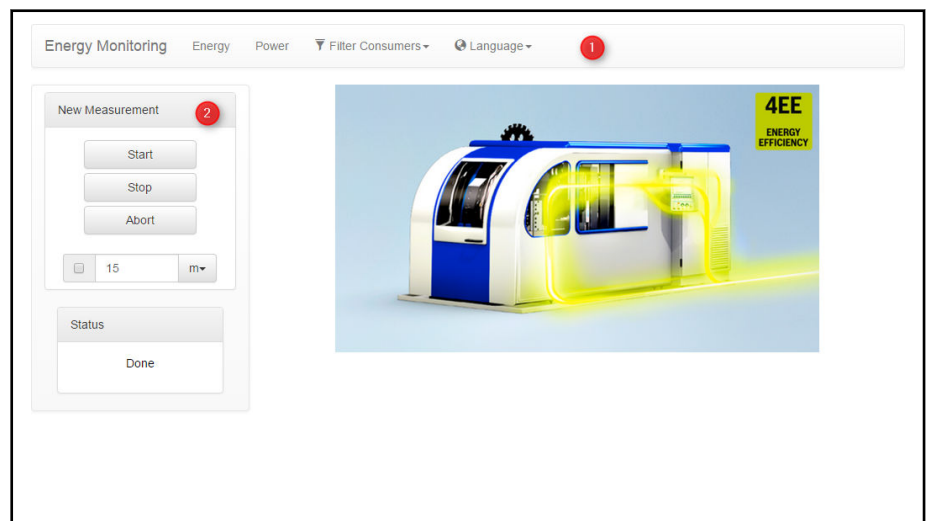
The update cycle of the performance display can be selected via the update time selection. By clicking on OK, the selected IP address and the update time are confirmed and the connection to the control is established.



- 1 Entering the IP address
- 2 Selecting the IP address
- 3 Selecting the refresh time
- 4 Connect button

Fig. 8-28: Connection window of the web-based visualization

The main screen is displayed after the connection to the control has been established successfully and after the configuration data has been read out of the interface structure. In this main screen, different screens can be selected to represent the energy data, the performance data, the filter options as well as the language selection via the navigation bar. The control panel to start, stop or cancel energy measurements is located at the left side.



- 1 Navigation bar
- 2 Control panel for energy measurement

Fig. 8-29: Main screen

Energy

Overview

The "Energy" screen is displayed via selection in the navigation bar. In this screen, the data of the last ten energy measurements are displayed in a bar diagram. These energy measurements are controlled via a control panel. To analyze the data, the ratio of individual consumers or consumer groups of the energy consumption are represented in pie or bar diagrams.

HMI Controls

Working with the energy monitor

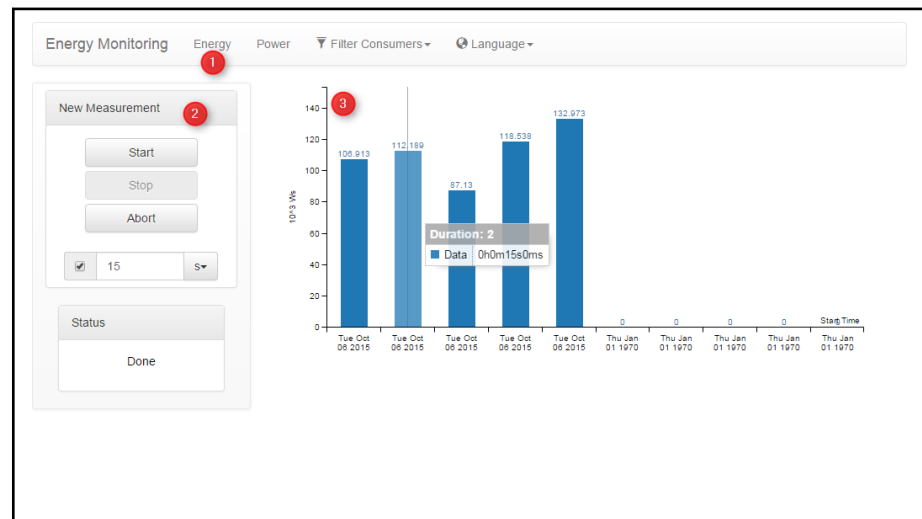
The control panel for energy measurements available in all screens exists of three buttons to start, stop and cancel the energy measurement, the input of a measurement duration and the status of the energy measurement. The measurement is always started via the topmost button. If the box for duration of the energy measurement is not ticked, a measurement via the second button is stopped manually. The third button is used to cancel a started measurement. Thus, the measured values are discarded and not displayed in the bar diagram.

Optionally to manually stopping the energy measurement can be executed with a fixed virtual time. Tick the time input box and enter the measurement duration. The unit (seconds, minutes or hours) can be selected via the drop-down menu. After starting the measurement, it is automatically stopped after the entered time has elapsed.



Starting and stopping the measurements is also possible via the HMIDataCollector function block interface in the PLC.

The status displays if a measurement is active or if the last measurement has been completed.



- 1 Selecting the screen
- 2 Control panel for energy measurement
- 3 Bar graph of energy measurements

Fig. 8-30: "Energy" screen

The measured data of the last then measurements are saved in the interface structure on the control. When opening the screen and updating the measurement data, the data is read out off the control and the total consumptions are displayed in the bar diagram. The duration of the individual measurements is displayed in the tooltip. By clicking on the measurements, a window to select the detailed view of the energy measurement opens. In the example of the following figure, the measurement data of the individual consumers and by selected via "All consumers". The measuring data are divided into the groups of the relevant grouping via "Device Type" or "Machine Module". Select between a pie (only for exclusively positive values) or a bar diagram.

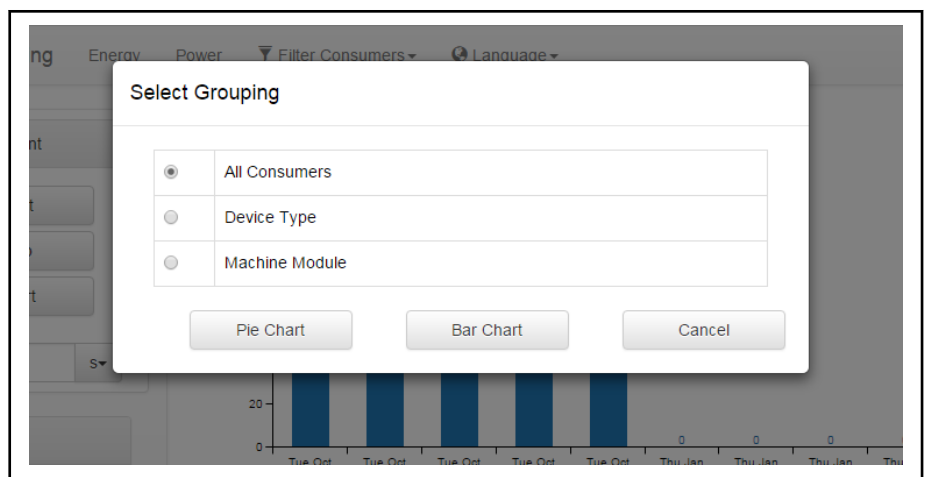


Fig. 8-31: Window to select a detailed view of an energy measurement

In the pie diagram, the percentage of consumers or consumer groups of the total consumption is displayed. The absolute value can be displayed in the tooltip. Click on the pie diagram or on the button on the right to return to the bar diagram.

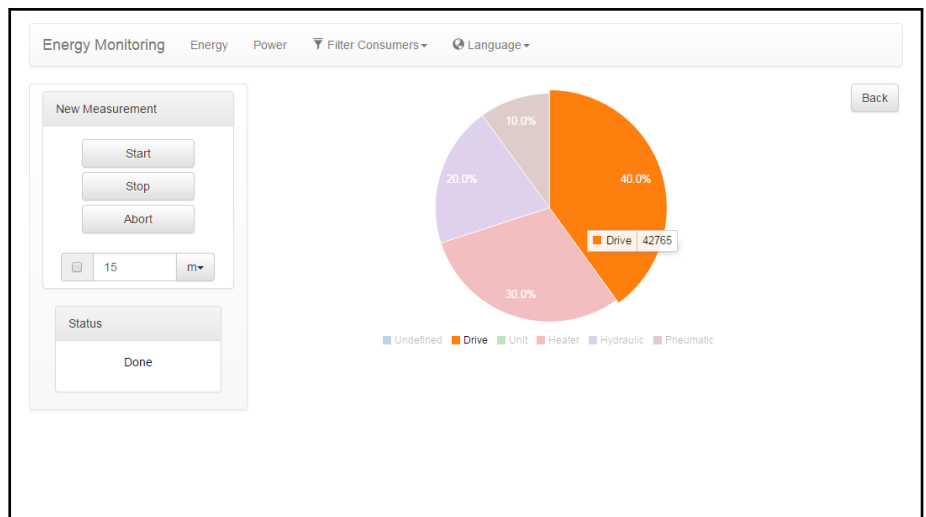


Fig. 8-32: Pie diagram of an energy measurement

Performance

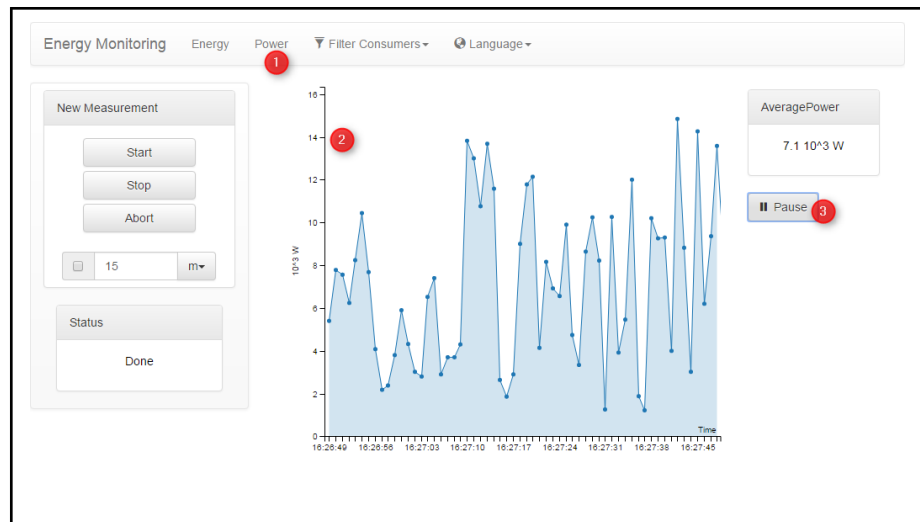
Overview

The "Power" screen is displayed via selection in the navigation bar. In this screen, the consumed power is displayed in a line diagram. The line diagram contains the last 60 measured values. Upon the start of the visualization, the sample time can be set in the connection window.

Working with the performance monitor

By clicking on the pause button, the recording can be stopped. The individual consumers or consumer groups can be displayed via the filter function ("Filter consumers").

HMI Controls



1 Selecting the screen
 2 Line diagram of the performance measurement
 3 Pause button
 Fig. 8-33: Line measurement of the performance measurement

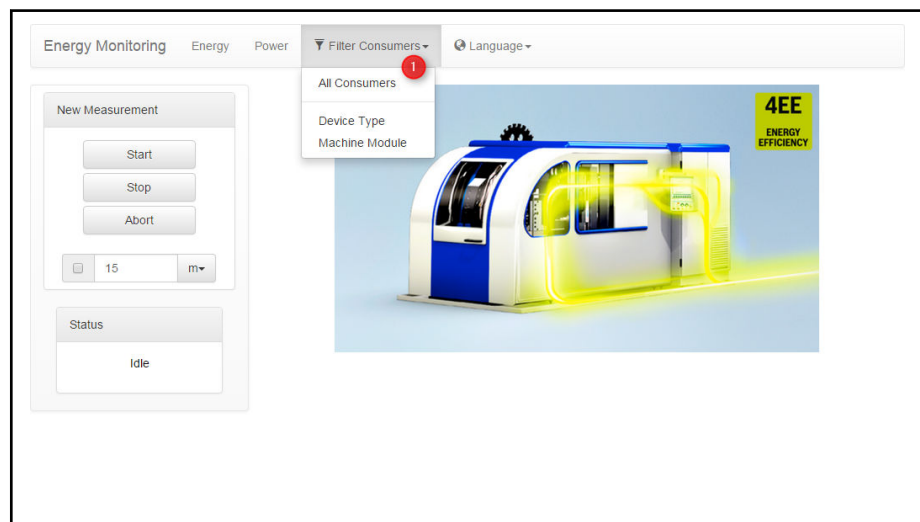
Filtering the consumers

Overview

Use the filter function to select which consumers or consumer groups are displayed in the energy and performance data displays. The screens to filter the consumers are selected via a dropdown menu in the navigation bar. A screen to select individual groups is now available for each used grouping apart from a table in which the consumers can be individually selected and deselected.

Working with the filter function

In the drop-down menu to select the filter screen, navigation to the screen to select individual consumers is possible via the first link. Above, you will find links to the screens of all groupings used in the PLC function block configuration.



1 Drop-down menu for the filter functions
 Fig. 8-34: Drop-down menu for the filter screens in the navigation

In the screen to select individual consumers, the filter functions and their group association are listed in a table. The individual consumers can be selected and deselected in the energy and performance data displays via the checkboxes in the first columns. All consumers can be simultaneously selec-

ted and deselected using the topmost checkbox. If at least one consumer is deselected, it is indicated in the navigation bar to the top right that a filter is active. An active filter is displayed on all screens until all consumers are deselected again.

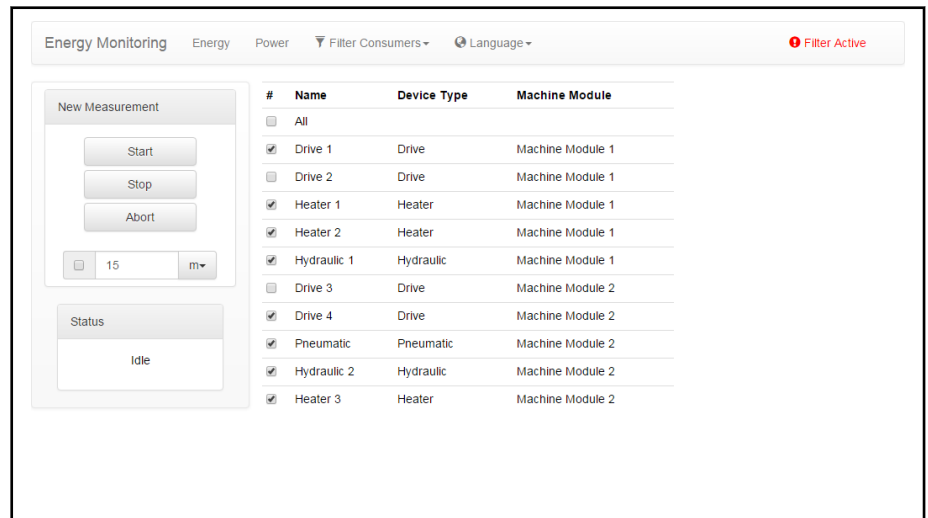


Fig. 8-35: Filter function: Selecting individual consumers

An individual group of the respective grouping can be selected in the screen to select consumer groups. All consumers of this group are selected and all consumers not part of the group are deselected. The group selection can be reset using the topmost radio button ("No Choice").

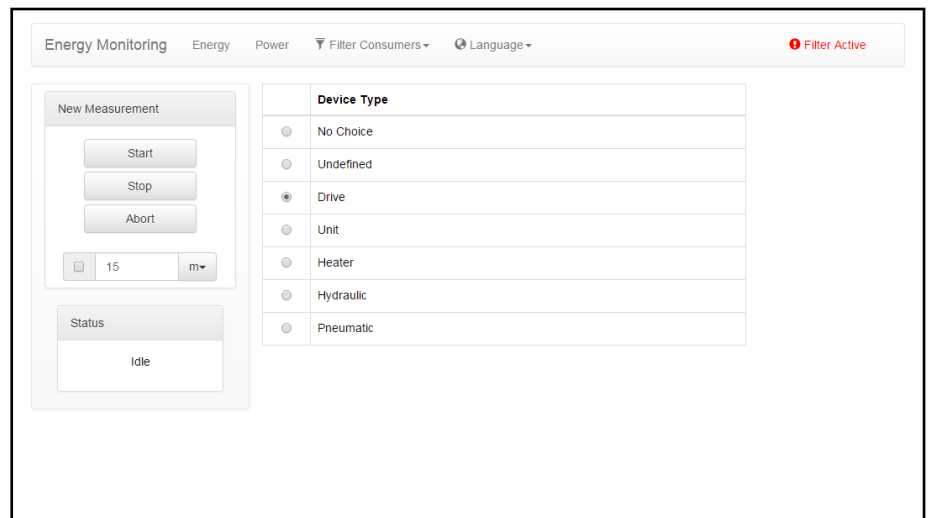


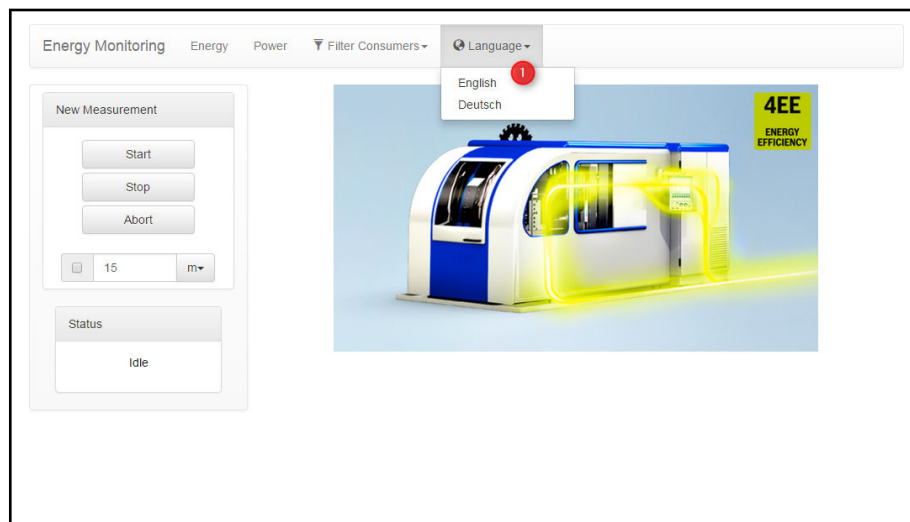
Fig. 8-36: Selecting the consumer groups

Language switching

Overview

Language switching is used to switch the texts in the web-based visualization. This comprises the application-specific texts that have to be specified by the user as well as the application-independent texts of the visualization. The [IL_4EEMonHMIDataCollectorType03, page 188](#) function block has to be used on the PLC. The texts are not read out of the PLC but of the "...\\WebConnector\\WEBROOT\\EnergyMonitoring\\json\\UserStrings.json" which has to be edited by the user. All application-dependent visualization texts are contained in German and English in "...\\WebConnector\\WEBROOT\\EnergyMonitoring\\json\\GuiStrings.json". This file has to be extended if additional languages are to be added.

HMI Controls

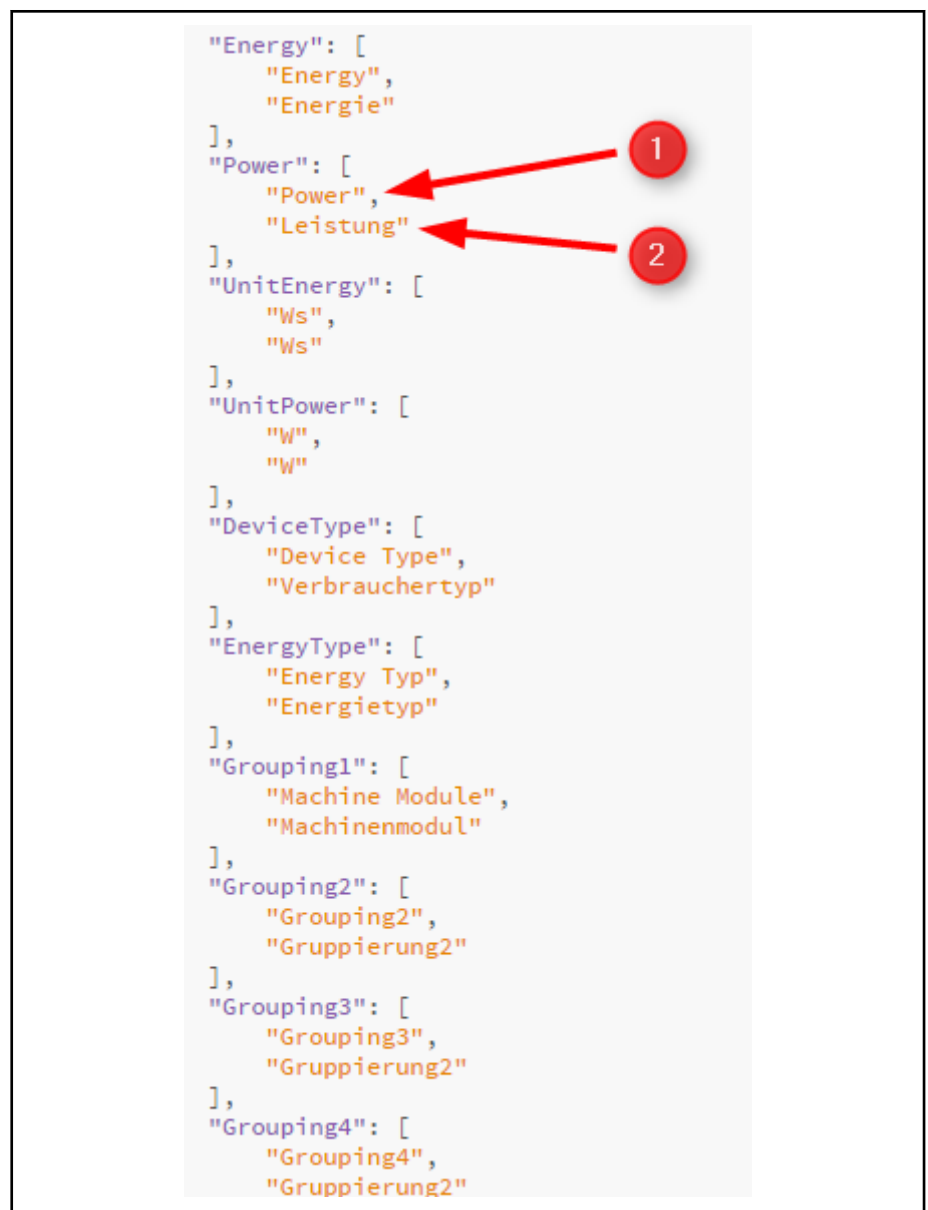


1 Drop-down menu for language switching

Fig. 8-37: Drop-down menu for language switching

Creating the configuration parameters for the user texts

The application-specific texts are configured via the "...\\WebConnector\\WEB-ROOT\\EnergyMonitoring\\json\\UserStrings.json" file. This file contains all strings configured at the function blocks on the PLC in case of no language switching. The file contains an object containing multiple arrays. This file can e.g. be edited with the programs Notepad or Notepad++. The first ten elements are simple string arrays containing texts for physical sizes, the units of physical sizes and the names of groupings. The first string is the English string, the second string is the German text. To add another language, a string has to be added to the arrays (also see ["Extending the configuration parameters for the visualization texts" on page 108](#)).

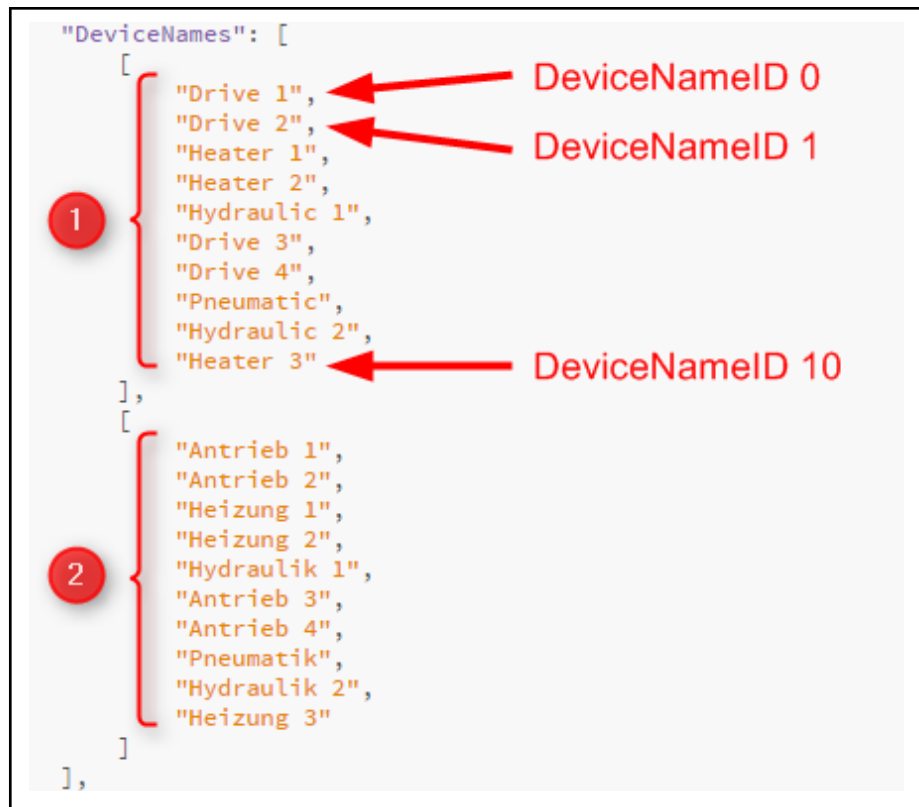


- 1 English text
 2 German text

Fig. 8-38: Simple arrays in the UserStrings.json file

Arrays of arrays are provided to name the consumers and the consumer groups. The elements of the external arrays correspond to the different languages. The strings in the internal arrays are the consumer names and the consumer group names. These strings are assigned to individual consumers via the values of the DeviceNameID, DeviceType, EnergyType and Grouping1 - Grouping4 inputs of the device function blocks on the PLC.

HMI Controls



- 1 English texts
2 German texts

Fig. 8-39: Array of arrays for the consumer names in the UserStrings.json file

Extending the configuration parameters for the visualization texts

The "...[WebConnector\WEBROOT\EnergyMonitoring\json\GuiStrings.json]" file contains all non-application specific texts of the web-based visualization. This configuration file only has to be edited if the visualization is to be extended by additional languages. The file contains an objects with both elements "Language" and "Strings". The "Language" element is an array containing the texts for the links in the drop-down menu of the language selection. The language sequence in this array also determines the language sequence in all other arrays of the GuiStrings.json and UserStrings.json files. By selecting the link "English" in the drop-down menu which has the index 0 in the "Language" object, the strings with this index (i.e., 0) are read out of all arrays.



Fig. 8-40: Array "Language" of the GuiStrings.json file

The element "Strings" is an array of objects. These objects contain the visualization texts (not application-specific), sorted according to the different screens.

```
"Strings": [{
  "MainPage": {
    "EnergyMonitoring": "Energy Monitoring",
    "FilterConsumers": "Filter Consumers",
    "AllConsumers": "All Consumers",
    "Language": "Language",
    "FilterActive": "Filter Active",
    "NewMeasurement": "New Measurement",
    "Start": "Start",
    "Stop": "Stop",
    "Abort": "Abort",
    "Status": "Status",
    "Idle": "Idle",
    "Active": "Active",
    "Done": "Done"
  },
  "Energy": {
    "SelectGrouping": "Select Grouping",
    "AllConsumers": "All Consumers",
    "PieChart": "Pie Chart",
    "BarChart": "Bar Chart",
    "Cancel": "Cancel",
    "Back": "Back",
    "Duration": "Duration"
  },
  "Power": {
    "AveragePower": "AveragePower",
    "Pause": "Pause"
  },
  "Filter": {
    "AllConsumers": "All Consumers",
    "Name": "Name",
    "NoChoice": "No Choice"
  }
}, {
  "MainPage": {
    "EnergyMonitoring": "EnergieMonitoring",
    "FilterConsumers": "Filter Verbraucher",
    "AllConsumers": "Alle Verbraucher",
    "Language": "Sprache",
    "FilterActive": "Filter Aktiv",
    "NewMeasurement": "Neue Messung",
    "Start": "Start",
    "Stop": "Stopp",
    "Abort": "Abbrechen",
    "Status": "Status",
```

1 English text
2 German text

Fig. 8-41: Array "Strings" of the GuiStrings.json file

9 RIL_4EEEnergyManagement.library

9.1 Overview

The EnergyManagement consists of the EnergyMonitoring for recording energy consumption, and the EnergyControl for controlling energy consumption by shutting off consumers.

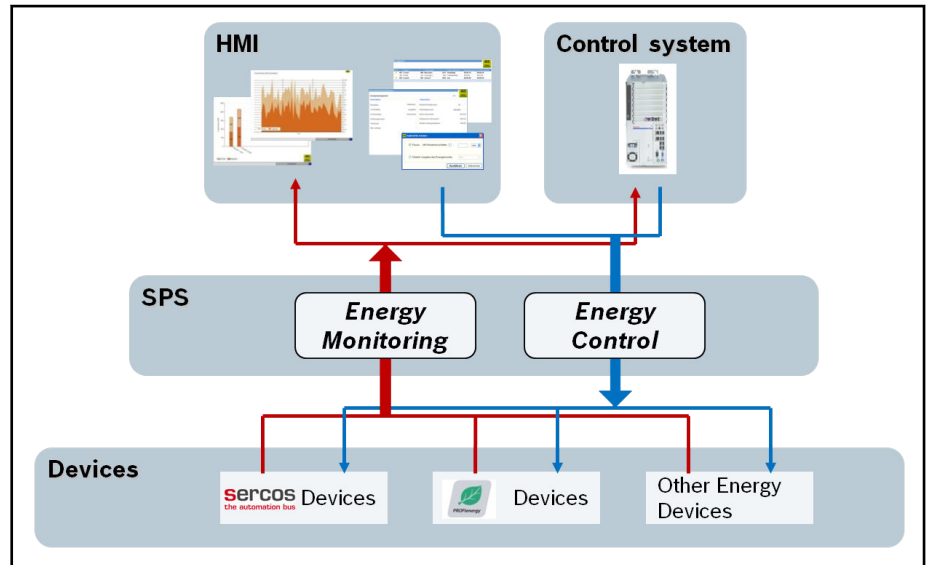


Fig. 9-1: EnergyManagement overview

Function blocks are found in the RIL_4EEEnergyManagement library for [EnergyMonitoring](#), p. 111, and [EnergyControl](#), p. 210. Linking the function blocks via object orientation mechanisms makes it possible to create an EnergyManagement from the consumer connection to the HMI connection.

9.2 EnergyMonitoring

9.2.1 Overview

The EnergyMonitoring is used to determine the energy consumption of the machine and its individual components. Function blocks are available that are linked as object-oriented. The instances of the device function blocks providing the central function block energy raw data are registered at a central EnergyMonitor function block. Different device function blocks for general consumers and for reading out the short time energy counter of IndraDrive drives are available.

The individual function blocks are divided into the following classes:

- EnergyMonitor
- MonitoringDevice
- DataCollector

An HMI collector function block is also registered at the central EnergyMonitor function block which uses the EnergyMonitor function block data for the interface structure to the HMI. This means that the relevant HMI collector function block is always required for HMI controls.

RIL_4EEEnergyManagement.library

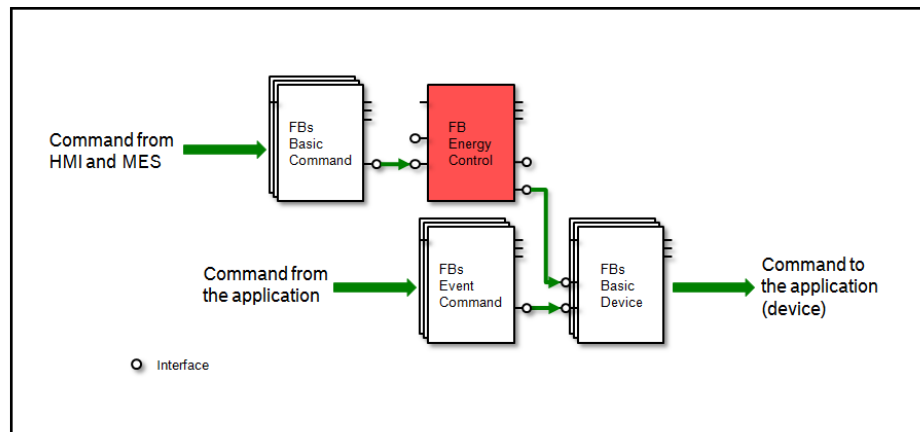


Fig. 9-2: EnergyMonitoring overview

The user can assign the “Device” function blocks to groups via the enumerations. One enumeration each is reserved for the device and energy type. The user can divide the consumer into application-specific groups using four inputs.

The energy data is cyclically queried by the central EnergyMonitor function block in an adjustable time. The EnergyMonitor provides the power and energy data to the HMI Collector function block which then sends this data to the HMI structure. Energy measurements can be started and stopped via the HMI Collector function block.

Function block	Brief description
IL_4EEEnergyMonitorType01, page 113	“Device” and “Collector” function blocks register at an instance of this block. The function block collects and calculates energy and power data and provides this data to the HMI
IL_4EEMonBasicDeviceType01, page 115	Function block for energy measurement in case of generic input of power or energy data
IL_4EEMonIndraDriveDeviceType01, page 120	Function block for measuring energy by reading the short-time energy counter for IndraDrive drives (P-0-0851, MPx 06-08 or from MPx 17)
IL_4EEMonSercosEnergyDeviceType01, page 124	Function block used for energy measurement in Sercos devices, supporting the Sercos Energy Monitoring functionality
IL_4EEMonProfiEnergyDeviceType01, page 128	Function block used for energy measurement in Sercos devices, supporting the PROFIenergy monitoring functionality
IL_4EEMonPowerSupplyDeviceType01, page 133	Function block for energy measurement to read out the energy counters of devices supporting the Sercos supplier profile (S-0-1702.0.14)
IL_4EEMonTerminalDeviceType01, page 137	Function block for energy measurement with power measurement terminals of Phoenix Contact
IL_4EEMonTerminalDeviceType02, page 142	Function block for energy measurement with power measurement terminals of Phoenix Contact and additional output of measuring objects
IL_4EEMonCompAirDeviceType01, page 148	Function block to generically gather energy and power data for consumers of compressed air
IL_4EEMonFluidDeviceType01, page 153	Function block to generically gather energy and power data for consumers of fluids and gases where energies are transmitted via temperature differences (e.g. cooling water, heating gas)

IL_4EEMonGasDeviceType01, page 158	Function block to generically gather energy and power data for consumers of combustible gases
IL_4EEMonGasDeviceType02, page 163	Function block to generically gather energy and power data for consumers of combustible gases
IL_4EEMonPneumaticDeviceType01, page 168	Function block to generically gather energy and power data of pneumatic cylinders
IL_4EEMonS0DeviceType01, page 174	Function block to analyze energy consumption values transmitted via a pulse interface acc. to EN 62053-31 (S0 interface)
IL_4EELoadProfileEstimation, page 178	Function block to create a load profile estimation on incoming energy consumption values
IL_4EEMonHMIDataCollectorType01, page 183	The collector function block copies the data to the HMI structure of the .NET controls and measures the energy
IL_4EEMonHMIDataCollectorType02, page 186	The collector function block copies the data to the HMI structure of the web visualization (without language switching) and measures the energy
IL_4EEMonHMIDataCollectorType03, page 188	The collector function block copies the data to the HMI structure of the web visualization (with language switching) and measures the energy
IL_4EEMonScadaDataCollectorType01, page 190	The collector function block copies the data to the MES/control system structure and measures the energy

Tab. 9-1: Overview of the EnergyMonitoring function blocks

9.2.2 Function blocks and their methods

IL_4EEEnergyMonitorType01

Brief description The instances of the monitoring device blocks are registered at the IL_4EEEnergyMonitor EnergyMonitor block. The IL_4EEEnergyMonitor queries the energy values from the registered device blocks and calculates the overall power/energy values used by the devices. HMI Collector function blocks are also registered on the block that copy the data in interface structures to the HMI controls.

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-2: Assignment: Target system/library

Interface description

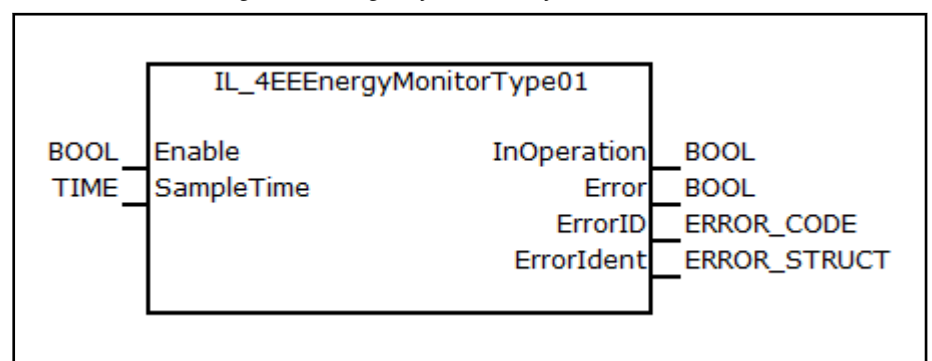


Fig. 9-3: IL_4EEEnergyMonitorType01 function block

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
	SampleTime	TIME	Indicates the sampling rate at which the energy values should be read
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Error handling, page 115

Tab. 9-3: IL_4EEEnergyMonitorType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
SampleTime	TIME			T#1s	Pos. edge "Enable"

Tab. 9-4: Minimum/maximum values and default values of the IL_4EEEnergyMonitorType01 inputs

Functional Description

The IL_4EEEnergyMonitor block is the central block of the EnergyMonitoring. The device blocks and HMICollector blocks are registered with an instance of the block.

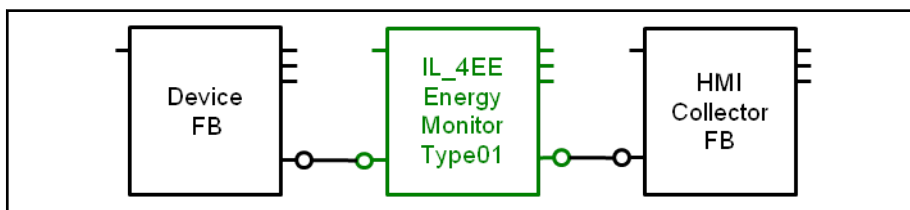


Fig. 9-4: Integrating IL_4EEEnergyMonitoringType01 into EnergyMonitoring

The blocks are connected via interfaces. In order to link the blocks together, the instance of the IL_4EEEnergyMonitorType01 with which the instances of the device and collector blocks should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the blocks can register and transmit the consumption values.

SYNTAX

```

VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, registered at the EnergyMonitor-FB *)
fbMonBasicDevice: IL_4EEMonBasicDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR

```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime. The block cyclically gathers the energy data from the devices and calculates the overall power. The cycle time in which the device blocks are queried can be configured through the SampleRate input. If a device block is configured as a total energy consumption counter (TotalConsumption input on device block), the difference in total energy consumption

and the sum of energy consumption for the other devices are generated and assigned to another "remaining energy" consumer:

$$W_{Rest} = W_{Gesamt} - \sum W_{Device}$$

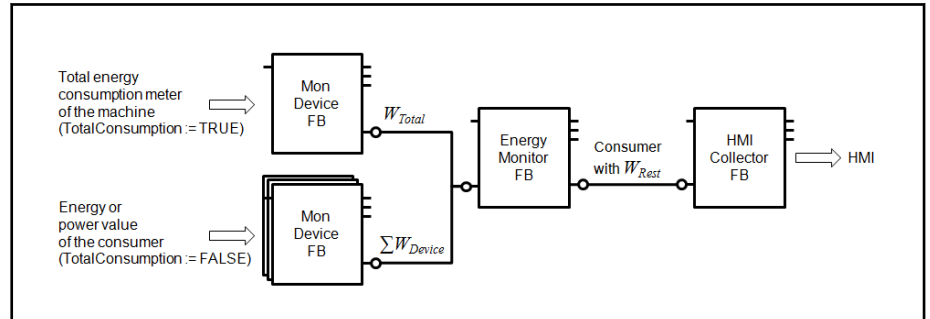


Fig. 9-5: Overview of calculating remaining energy consumption

Only one device block can be registered as the total energy consumption counter.

Error handling

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
STATE_MACHINE_ERROR (16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-5: IL_4EEEnergyMonitorType01 error codes

IL_4EEMonBasicDeviceType01

Brief description

The IL_4EEMonBasicDeviceType01 device block serves to generically collect energy and performance data for general consumers. The block is registered in an EnergyMonitor block. Energy or performance data can be used for input data.

The function block provides the following methods as an interface:

- mSetEnergyOdometer
- mSetPowerConsumption
- mSetConsumerActive

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-6: Assignment target system/library

RIL_4EEEnergyManagement.library

Interface description

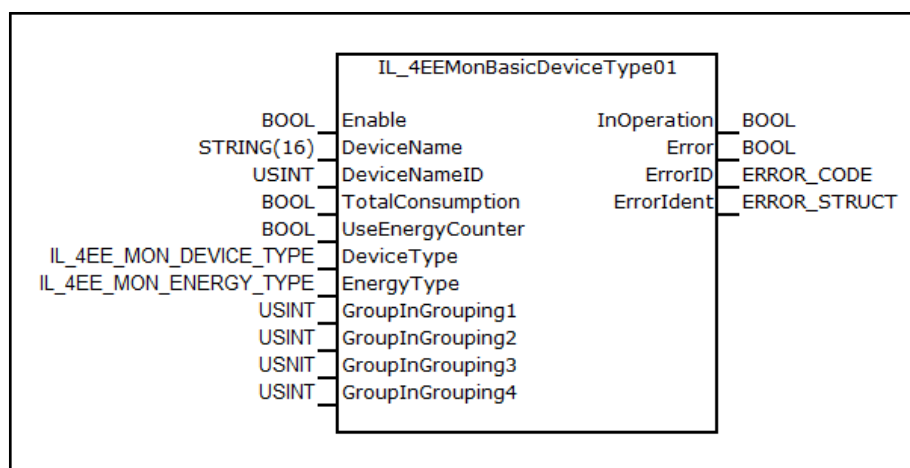


Fig. 9-6: IL_4EEMonBasicDeviceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	DeviceName	STRING(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name, used for display on HMIS with language switching
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	UseEnergyOdometer	BOOL	Select energy counter with TRUE. Otherwise performance data is used.
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Select device type, used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	EnergyType	IL_4EE_MON_ENERGY_TYPE	Select energy type, used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1, used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2, used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3, used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4, used for HMI display
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostic, see Troubleshooting , page 119

Tab. 9-7: IL_4EEMonBasicDeviceType01 function block

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. Edge Enable
DeviceNameID	USINT			0	Pos. Edge Enable
TotalConsumption	BOOL			FALSE	Pos. Edge Enable
UseEnergyCounter	BOOL			FALSE	Pos. Edge Enable
DeviceType	IL_4EE_MON_DE- VICE_TYPE	DEV_TYPE_UNDE- FINED	DEV_TYPE_USER_ DEFINED_63	DEV_TYPE_UNIT	Pos. Edge Enable
EnergyType	IL_4EE_MON_ENER- GY_TYPE	E_TYPE_UNDE- FINED	E_TYPE_USER_DE- FINED_31	E_TYPE_ELEC- TRICAL	Pos. Edge Enable
GroupInGrouping1	USINT	0	63	0	Pos. Edge Enable
GroupInGrouping2	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping3	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping4	USINT	0	31	0	Pos. Edge Enable

Tab. 9-8: Minimum/maximum and default values of the IL_4EEMonBasicDeviceType01 inputs

Block Methods

The function block provides the following methods as an interface:

- [mSetEnergyOdometer, page 117](#)
- [mSetPowerConsumption, page 117](#)
- [mSetConsumerActive, page 118](#)

mSetEnergyOdometer

The mSetEnergyOdometer method serves to set the current value of the energy counter for the consumer in the block. This method is only relevant if the value TRUE was selected for the block's UseEnergyOdometer input and the block is in the InOperation state.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	EnergyOdometerValue	REAL			Current value of the energy counter in [W]

Tab. 9-9: Interface for the mSetEnergyOdometer method with min./max. values

Both positive and negative values are possible for the energy counter. Make sure that the energy counter should behave according to a DINT variable in regard to overflow. Therefore only a count range between $2^{31}-1$ and -2^{31} is valid. Otherwise the block enters an error state.

mSetPowerConsumption

The mSetPowerConsumption method serves to set the current power consumption for the consumer in the block. This method is only relevant if the value FALSE was selected for the block's UseEnergyOdometer input and the block is in the InOperation state.

RIL_4EEEnergyManagement.library

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	PowerConsumptionValue	REAL			Current power consumption of consumer in [W]

Tab. 9-10: Interface for the mSetPowerConsumption method with min./max. values

mSetConsumerActive The mSetConsumerActive method serves to notify the block during power measurements that the consumer is off. The device block then stops the energy measurement (power value = 0 W). When initializing the block, it is assumed that the consumer is on. This method is only relevant if the value FALSE was selected for the block's UseEnergyOdometer input and the block is in the InOperation state.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	ConsumerActive	BOOL			TRUE: The consumer is on, energy is measured with the last value from the mSetPowerConsumption method FALSE: The consumer is off, energy measurement stops

Tab. 9-11: Interface for the mSetConsumerActive method with min./max. values

Functional description The IL_4EEMonBasicDeviceType01 function block registers on a central EnergyMonitor block during instantiation.

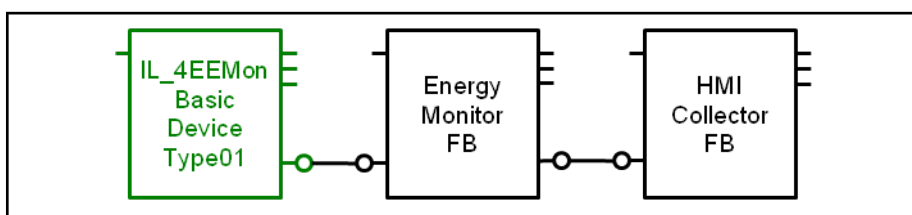


Fig. 9-7: Integrating IL_4EEMonBasicDeviceType01 into EnergyMonitoring

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyMonitor block with which the instance of the IL_4EEMonBasicDeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```

VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonBasicDevice: IL_4EEMonBasicDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR

```

Maximum Number of Blocks Registering

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

Configuration

The number of device blocks that can be registered with an EnergyControl block can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in [Parameter list, page 297](#). The default value is 84.

The UseEnergyOdometer input can select the type of input data. On the one hand, the energy counter values can be entered directly via the mSetEnergyOdometer method. These are then queried by the EnergyMonitor block.

There are two ways to specify power values:

- Specify the current output power via the mSetPowerConsumption method. When changing the power consumption of the device, this is changed by calling the method again.
- Specify the average output power via the mSetPowerConsumption method. The mSetConsumerActive method notifies the block whether or not the device is on or off. When on, the average output power is used to calculate the energy; when off, this value is set internally to 0.

The TotalConsumption input can configure the block as the total power consumption counter for the machine.

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be less than 64

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0006	Cause: Energy value outside valid range between $(2^{31})-1$ and $2^{(-31)}$ Solution: The external energy counter value must remain within the limits
OTHER_ERROR (16#7FFE)	16#2A02	16#0000	Cause: The block could not be registered with the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR (16#7FFE)	16#2A02	16#0001	Cause: The device block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one device block can be registered as the TotalConsumption device
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-12: Error codes of the IL_4EEMonBasicDeviceType01 function block

IL_4EEMonIndraDriveDeviceType01

Brief description The IL_4EEMonIndraDriveDeviceType01 device block serves to measure the energy for an IndraDrive drive with short-time energy counter. The block reads the parameter P-0-0851 and transmits it to the EnergyMonitor block to which it is registered.

The function block provides the following method as an interface:

- mSetAdditivePowerConsumption

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-13: Assignment target system/library

Interface description

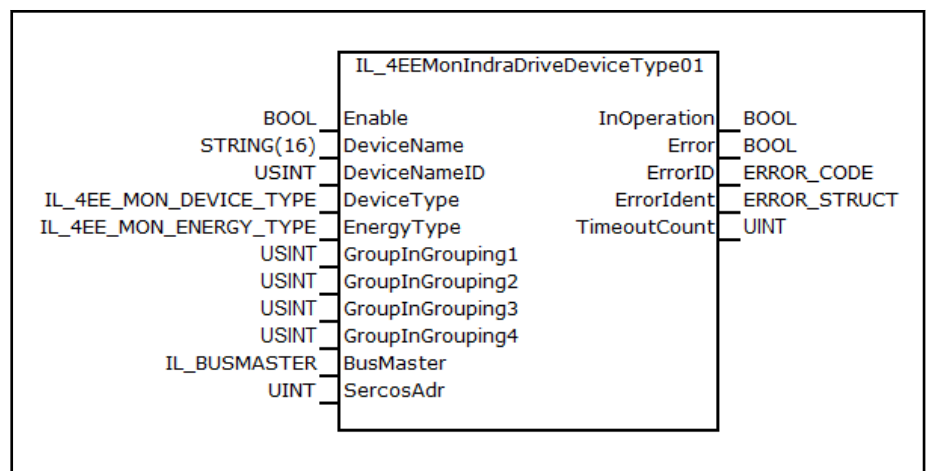


Fig. 9-8: IL_4EEMonIndraDriveDeviceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	DeviceName	STRING(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name, used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Select device type, used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	EnergyType	IL_4EE_MON_ENERGY_TYPE	Select energy type, used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1, used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2, used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3, used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4, used for HMI display
	BusMaster	IL_BUSMASTER	Sercos master ID. If simultaneously operated on the control system, multiple Sercos masters are distinguished by the BusMaster input variable. If operated alone, a single Sercos master has master ID = IL_BUSMASTER_0.
	SercosAdr	UINT	Sercos III address of the slave device
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Troubleshooting, page 123
	TimeoutCount	UINT	Number of timeouts occurred for parameter reading

Tab. 9-14: IL_4EEMonIndraDriveDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. Edge Enable
DeviceNameID	USINT			0	Pos. Edge Enable
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_DRIVE	Pos. Edge Enable
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. Edge Enable
GroupInGrouping1	USINT	0	63	0	Pos. Edge Enable
GroupInGrouping2	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping3	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping4	USINT	0	31	0	Pos. Edge Enable
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Pos. Edge Enable
SercosAdr	UINT	1	511	0	Pos. Edge Enable

Tab. 9-15: Minimum/maximum and default values of the IL_4EEMonIndraDriveDeviceType01 inputs

Block Methods The function block provides the following methods as an interface:

- [mSetAdditivePowerConsumption, page 122](#)

mSetAdditivePowerConsumption

The mSetAdditivePowerConsumption method serves to record the additional power consumption of the consumer, such as 24 V output power of the control unit or the energy consumption of a holding brake. Specify the current value using the method. The block integrates the power value and adds it to the measured value read from the drive.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	AdditivePowerConsumption-Value	REAL			Current additional power consumption of consumer in [W]

Tab. 9-16: Interface for the mSetAdditivePowerConsumption method with min./max. values

Functional description The IL_4EEMonIndraDriveDeviceType01 function block registers on a central EnergyMonitor block during instantiation.

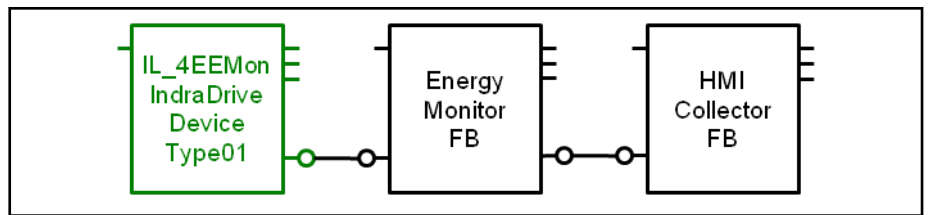


Fig. 9-9: Integrating IL_4EEMonIndraDriveDeviceType01 into EnergyMonitoring

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyMonitor block with which the instance of the IL_4EEMonIndraDriveDeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonBasicDevice: IL_4EEMonIndraDriveDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);;
END_VAR
```

Maximum Number of Blocks Registering

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

Configuration

The number of device blocks that can be registered at an EnergyControl block can be modified with the constant "NO_OF_MON_DEVICES_PER_ENERGY_MONITOR" in [Parameter list, page 297](#). The default value is 84.

The Sercos address for the axis is indicated by the BusMaster and SercosAdr inputs whose energy data are to be read out.

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

The TimeoutCount output increases by one in the event of a read block timeout. If this output increases frequently, either the SampleTime for the EnergyMonitor block can be increased or the parameter P-0-0851 can be entered in the cyclical Sercos channel of the corresponding drive.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2 in addition to IL_SIIISvcRead function block error messages:

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0007	Cause: BusMaster input has no valid value Solution: The input has to have the value IL_BUSMASTER_0
INPUT_RANGE_ERROR	16#2A00	16#0008	Cause: SercosAdr input not in valid range Solution: The SercosAdr input has to have a value between 1 and 511
RESOURCE_ERROR	16#2A01	16#0000	Cause: Parameter P-0-0851 (short-time energy counter) does not exist for device Solution: The function block can not be used for the drive
OTHER_ERROR	16#2A02	16#0000	Cause: The block could not be registered with the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-17: Error codes of the IL_4EEMonIndraDriveDeviceType01 function block

IL_4EEMonSercosEnergyDeviceType01

Brief description The device function block IL_4EEMonSercosEnergyDeviceType01 is used to read out energy measurement data from Sercos components, supporting Sercos Energy Monitoring functions. The energy consumption or load capacity is read out via these energy measurement data.

RIL_4EEEnergyManagement.library

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-18: Assignment target system/library

Interface description

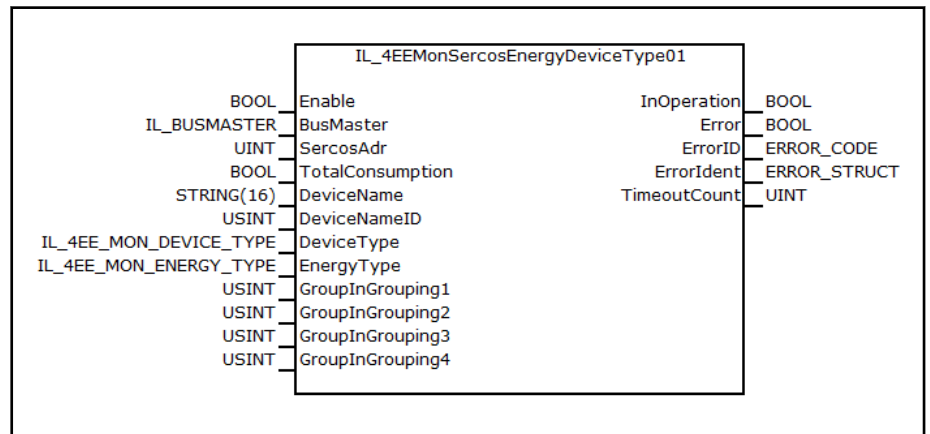


Fig. 9-10: IL_4EEMonSercosEnergyDeviceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	BusMaster	IL_BUSMASTER	Sercos master ID. If operated simultaneously on the control system, multiple Sercos masters are distinguished by the BusMaster input variable. If operated alone, a single Sercos master has the master ID = IL_BUSMASTER_0.
	SercosAdr	UINT	Sercos III address of the slave device
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	DeviceName	STRING(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name, used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Select device type, used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	EnergyType	IL_4EE_MON_ENERGY_TYPE	Select energy type, used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1, used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2, used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3, used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4, used for HMI display

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Troubleshooting, page 127
	TimeoutCount	UINT	Number of timeouts occurred for parameter reading

Tab. 9-19: IL_4EEMonSercosEnergyDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Pos. Edge Enable
SercosAdr	UINT	1	511	0	Pos. Edge Enable
TotalConsumption	BOOL			FALSE	Pos. Edge Enable
DeviceName	STRING(16)			"	Pos. Edge Enable
DeviceNameID	USINT			0	Pos. Edge Enable
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_DRIVE	Pos. Edge Enable
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. Edge Enable
GroupInGrouping1	USINT	0	63	0	Pos. Edge Enable
GroupInGrouping2	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping3	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping4	USINT	0	31	0	Pos. Edge Enable

Tab. 9-20: Min./max. and default values of the IL_4EEMonSercosEnergyDeviceType01 inputs

Functional description

The IL_4EEMonIndraDriveDeviceType01 function block registers on a central EnergyMonitor block during the instantiation.

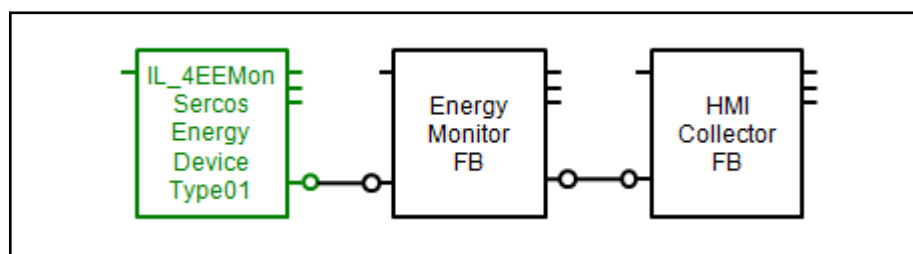


Fig. 9-11: Integrating IL_4EEMonSercosEnergyDeviceType01 into Energy Monitoring

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyMonitor block with which the instance of the IL_4EEMonSercosEnergyDeviceType01 should be registered has to be

specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonSercosEnergyDevice: IL_4EEMonSercosEnergyDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR
```

Maximum Number of Blocks Registering

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

The number of device blocks that can be registered with an EnergyControl block can be modified with the constant "NO_OF_MON_DEVICES_PER_ENERGY_MONITOR" in [Parameter list, page 297](#). The default value is 84.

Configuration

The Sercos address of the Sercos components whose energy data are to be read is specified by the BusMaster and SercosAdr inputs.

The TotalConsumption input can configure the block as the total power consumption counter for the machine. An example is a energy measuring device, equipped with a Sercos interface, supporting Sercos Energy.

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

The TimeoutCount output increases by one in the event of a read block timeout. If this output increases frequently, either the SampleTime for the EnergyMonitor block can be increased or the parameters S-0-1323.0.7 Consumed Energy Odometer or S-0-1323.0.8 Supplied Energy Odometer can be entered in the cyclical Sercos channel of the corresponding drive.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2 in addition to IL_SIIIsvRead function block error messages:

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0007	Cause: BusMaster input has no valid value Solution: The input has to have the value IL_BUSMASTER_0
INPUT_RANGE_ERROR	16#2A00	16#0008	Cause: SercosAdr input not in valid range Solution: The SercosAdr input has to have a value between 1 and 511
RESOURCE_ERROR	16#2A01	16#0001	Cause: The parameters S-0-1323.0.7 Consumed Energy Odometer and S-0-1323.0.8 Supplied Energy Odometer are available in the device Solution: The function block cannot be used for the device
OTHER_ERROR	16#2A02	16#0000	Cause: The block could not be registered with the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR	16#2A02	16#0001	Cause: The device block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one device block can be registered as the TotalConsumption device
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-21: Error codes IL_4EEMonSercosEnergyDeviceType01 function block

IL_4EEMonProfiEnergyDeviceType01

Brief description The device function block IL_4EEMonProfiEnergyDeviceType01 is used to read out energy measurement data from Profinet components, supporting

RIL_4EEEnergyManagement.library

PROFIenergy monitoring functions. Moreover, the function block is used to feed in data in Energy monitoring. Communication with the PROFIenergy device is executed via the IL_4EEProfiEnergyServiceType01 function block.

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-22: Assignment target system/library

Interface description

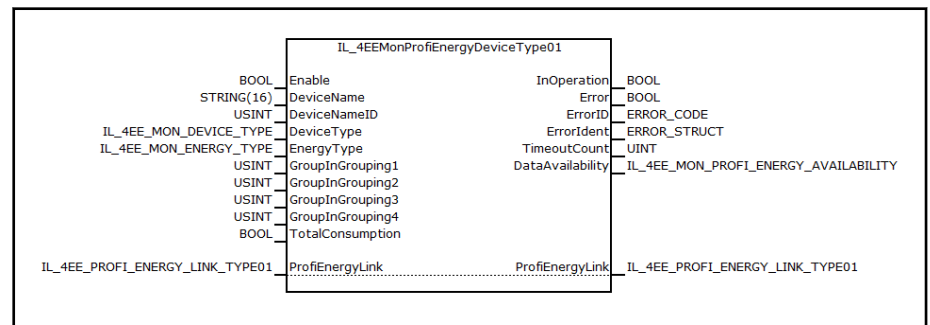


Fig. 9-12: IL_4EEMonProfiEnergyDeviceType01 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	ProfiEnergy-Link	IL_4EE_PROFI_ENERGY_LINK	A handle to PROFIenergy services is provided by the IL_4EEProfiEnergyServiceType01 function block
VAR_INPUT	Enable	BOOL	Processing enabled for function block (state-controlled)
	DeviceName	String(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name, used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Select device type, used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	EnergyType	IL_4EE_MON_ENERGY_TYPE	Select energy type, used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1, used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2, used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3, used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4, used for HMI display
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
VAR_OUT-PUT	InOperation	BOOL	Error-free operation; output data is valid

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	Error	BOOL	Error diagnostics
	ErrorID	ERROR_CODE	Detailed error diagnostics
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Troubleshooting, page 132
	DataAvailability	IL_4EE_MON_PROFI_ENERGY_AVAILABILITY	Status display, indicating which measured values are read See IL_4EE_MON_PROFI_ENERGY_AVAILABILITY, page 195

Tab. 9-23: IL_4EEMonProfiEnergyDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Pos. Edge at Enable
DeviceName	String(16)			"	Pos. Edge at Enable
DeviceNameID	USINT			0	Pos. Edge at Enable
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. Edge at Enable
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. Edge at Enable
GroupInGrouping1	USINT	0	63	0	Pos. Edge at Enable
GroupInGrouping2	USINT	0	31	0	Pos. Edge at Enable
GroupInGrouping3	USINT	0	31	0	Pos. Edge at Enable
GroupInGrouping4	USINT	0	31	0	Pos. Edge at Enable
TotalConsumption	BOOL			FALSE	Pos. Edge at Enable

Tab. 9-24: Min./max. and default values of the IL_4EEMonProfiEnergyDeviceType01 inputs

Functional description The IL_4EEMonProfiEnergyDeviceType01 function block registers on a central EnergyMonitor function block during the instantiation.

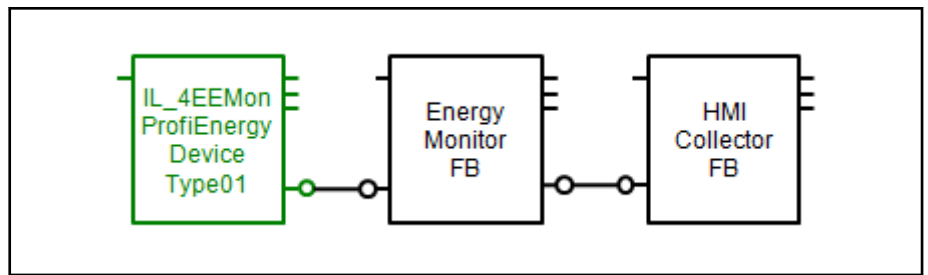


Fig. 9-13: Integrating IL_4EEMonProfiEnergyDeviceType01 into EnergyMonitoring

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyMonitor block with which the instance of the IL_4EEMonPofiEnergyDeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonSercosEnergyDevice: IL_4EEMonProfiEnergyDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

Furthermore, the IL_4EEMonProfiEnergyDeviceType01 has to be linked to an instance of the IL_4EEProfiEnergyServiceType01 function block. This function block is responsible for the communication with the Profinet device. This is required to be able to use an EnergyControl function block for the device (PROFenergy is not multi-client-compatible). Linking is realized via the ProfiEnergyLink handle. The output of the IL_4EEProfiEnergyServiceType01 function block has to be applied to the input/output of the same name of the IL_4EEMonProfiEnergyDeviceType01 function block.

Maximum Number of Blocks Registering

The number of device blocks that can be registered at an EnergyControl block can be modified with the constant "NO_OF_MON_DEVICES_PER_ENERGY_MONITOR" in [Parameter list, page 297](#). The default value is 84.

Configuration

The Profinet device selection is configured via the inputs of the IL_4EEProfiEnergyServiceType01 function block.

The TotalConsumption input can configure the block as the total power consumption counter for the machine. An example is an energy measuring device, equipped with a Profinet interface, supporting Sercos Energy.

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

RIL_4EEEnergyManagement.library

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

The output DataAvailability shows which measured values can be read out by the function block from the PROFIenergy device.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. The function block can generate the following error messages in Additional1/Additional2, apart from the error messages of the IL_4EEProfiEnergyService-Type01 and IL_PNIOServiceRecord function blocks:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be less than 32
RESOURCE_ERROR	16#2A01	16#0002	Cause: The PROFIenergy measuring functions are not implemented in the device Solution: The device cannot be read with this function block
RESOURCE_ERROR	16#2A01	16#0003	Cause: None of the required measurement Ids is available in the device Solution: The device cannot be read with this function block
OTHER_ERROR	16#2A02	16#0000	Cause: The block could not be registered with the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list

ErrorID	Additional1	Additional2	Description
OTHER_ERROR	16#2A02	16#0001	Cause: The device block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one device block can be registered as the TotalConsumption device
OTHER_ERROR	16#2A02	16#0006	Cause: No valid PROFIenergyLink was created Solution: Apply output ProfiEnergyLink of the function block IL_4EEProfiEnergyServiceType01 at the ProfiEnergyLink input/output.
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-25: Error codes function block IL_4EEMonProfiEnergyDeviceType01

IL_4EEMonPowerSupplyDeviceType01

Brief description The device function block IL_4EEMonPowerSupplyDeviceType01 is used to read out devices supporting the Sercos supplier profile. The block reads the parameter S-0-1702.0.14 and transmits it to the EnergyMonitor block at which it is registered.

The function block provides the following method as an interface:

- mSetAdditivePowerConsumption

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-26: Assignment target system/library

Interface description

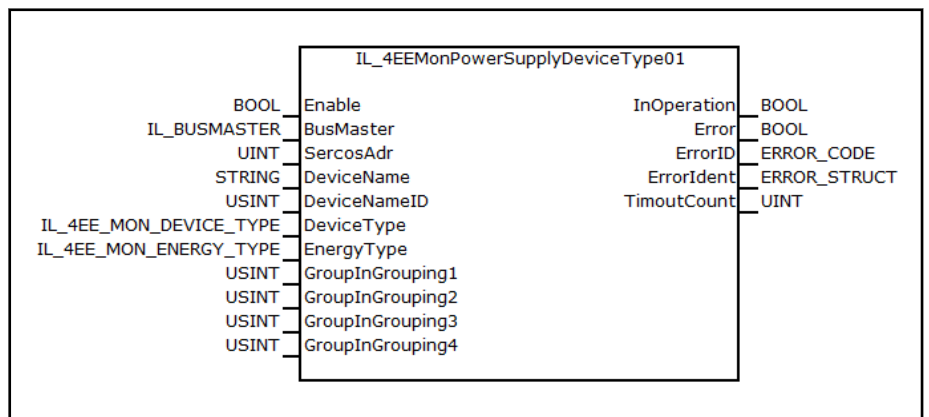


Fig. 9-14: IL_4EEMonPowerSupplyDeviceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	BusMaster	IL_BUSMASTER	Sercos master ID. If operated simultaneously on the control system, multiple Sercos masters are distinguished by the BusMaster input variable. If operated alone, a single Sercos master has the master ID = IL_BUSMASTER_0.
	SercosAdr	UINT	Sercos III address of the supplier
	DeviceName	STRING(16)	Name of device, used for HMI display

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	DeviceNameID	USINT	ID of the device name, used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Select device type, used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	EnergyType	IL_4EE_MON_ENERGY_TYPE	Select energy type, used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1, used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2, used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3, used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4, used for HMI display
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Troubleshooting , page 136
	TimeoutCount	UINT	Number of timeouts occurred for parameter reading

Tab. 9-27: IL_4EEMonPowerSupplyDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_0	IL_BUSMASTER_0	Pos. Edge Enable
SercosAdr	UINT	1	511	0	Pos. Edge Enable
DeviceName	STRING(16)			"	Pos. Edge Enable
DeviceNameID	USINT			0	Pos. Edge Enable
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. Edge Enable
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. Edge Enable
GroupInGrouping1	USINT	0	63	0	Pos. Edge Enable
GroupInGrouping2	USINT	0	31	0	Pos. Edge Enable

Name	Type	Min. value	Max. value	Default value	Effective
GroupInGrouping3	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping4	USINT	0	31	0	Pos. Edge Enable

Tab. 9-28: Min./max. and default values of the IL_4EEMonPowerSupplyDeviceType01 inputs

Block Methods The function block provides the following methods as an interface:

- [mSetAdditivePowerConsumption, page 135](#)

mSetAdditivePowerConsumption The mSetAdditivePowerConsumption method serves to record the additional power consumption of the consumer, such as 24V of the control unit. Specify the current value using the method. The block integrates the power value and adds it to the measured value read from the drive.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	AdditivePower-Consumption-Value	REAL			Current additional power consumption of consumer in [W]

Tab. 9-29: Interface for the mSetAdditivePowerConsumption method with min./max. values

Functional description The IL_4EEMonHMIDataCollectorType01 function block registers on a central EnergyMonitor block during instantiation.

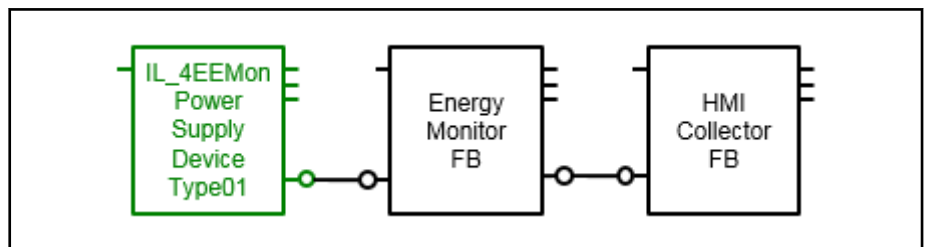


Fig. 9-15: Integrating IL_4EEMonProfiEnergyDeviceType01 into Energy Monitoring

The blocks are connected via interfaces. In order to link the function blocks, the instance of the EnergyMonitor block at which the instance of the IL_4EEMonHMIDataCollectorType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonPowerSupplyDevice: IL_4EEMonPowerSupplyDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);;
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

RIL_4EEEnergyManagement.library

Maximum Number of Blocks Registering

The number of device blocks that can be registered at an EnergyControl block can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in [Parameter list, page 297](#),. The default value is 84.

Configuration

The Sercos address of the supplies is indicated by the BusMaster and SercosAdr inputs whose energy data is to be read out.

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

The TimeoutCount output increases by one in the event of a read block time-out. If this output increases frequently, either the SampleTime for the Energy-Monitor block can be increased or the parameter S-0-1702.0.14 can be entered in the cyclical Sercos channel of the corresponding drive.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2 in addition to IL_SIIISvcRead function block error messages:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be less than 32

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A00	16#0007	Cause: BusMaster input has no valid value Solution: The input has to have the value IL_BUSMASTER_0
INPUT_RANGE_ERROR	16#2A00	16#0008	Cause: SercosAdr input not in valid range Solution: The SercosAdr input has to have a value between 1 and 511
RESOURCE_ERROR	16#2A01	16#0004	Cause: Parameter S-0-1702.0.14 does not exist in the device Solution: The function block can not be used for the supplier
OTHER_ERROR	16#2A02	16#0000	Cause: The block could not be registered with the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-30: IL_4EEMonPowerSupplyDeviceType01 function block error codes

IL_4EEMonTerminalDeviceType01

Brief description The device function block IL_4EEMonTerminalDeviceType01 is used to read out the inline energy measurement terminal IB IL PM 3P/N/EF-2MBD-PAC - 2700576 via the bus couplers for Sercos (IL S3 BK DI8 DO4 2TX-PAC - 2692380) and Profinet (IL PN BK DI8 DO4 2TX-PAC - 2703994) of Phoenix Contact. The function block is registered at an EnergyMonitor function block to which it transmits the energy data.

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-31: Assignment target system/library

Interface description

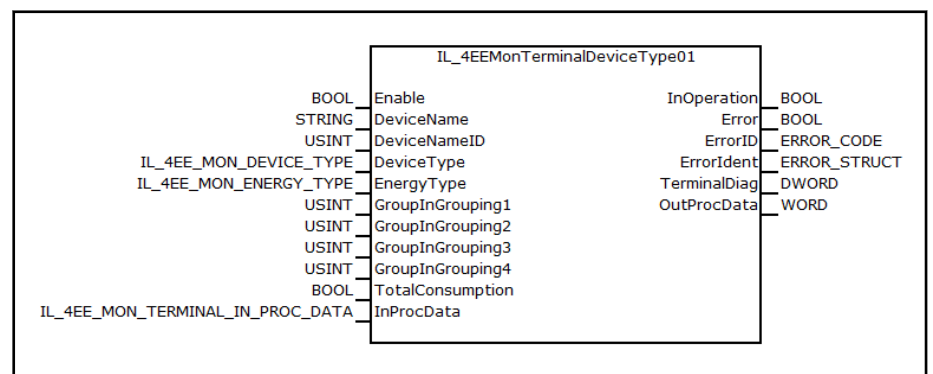


Fig. 9-16: IL_4EEMonTerminalDeviceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	DeviceName	STRING(16)	Name of device, used for HMI display

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	DeviceNameID	USINT	ID of the device name, used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Select device type, used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	EnergyType	IL_4EE_MON_ENERGY_TYPE	Select energy type, used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1, used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2, used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3, used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4, used for HMI display
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	InProcData	IL_4EE_MON_TERMINAL_IN_PROC_DATA	Cyclic input data of the energy measurement terminal See IL_4EE_MON_TERMINAL_IN_PROC_DATA , page 196
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Troubleshooting , page 141
	TerminalDiag	DWORD	Bit-coded output of terminal diagnostics
	OutProcData	WORD	Cyclic output data of the energy measurement terminal contains the control word to select the measuring object

Tab. 9-32: IL_4EEMonTerminalDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. Edge Enable
DeviceNameID	USINT			0	Pos. Edge Enable
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. Edge Enable
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. Edge Enable

Name	Type	Min. value	Max. value	Default value	Effective
GroupInGrouping1	USINT	0	63	0	Pos. Edge Enable
GroupInGrouping2	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping3	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping4	USINT	0	31	0	Pos. Edge Enable
TotalConsumption	BOOL			FALSE	Pos. Edge Enable
InProcData	IL_4EE_MON_TERMINAL_IN_PRO C_DATA				Continuous

Tab. 9-33: Min./max. and default values of the S-0-1702.0.14 inputs

Functional description

The IL_4EEMonTerminalDeviceType01 function block registers at a central EnergyMonitor function block during instantiation.

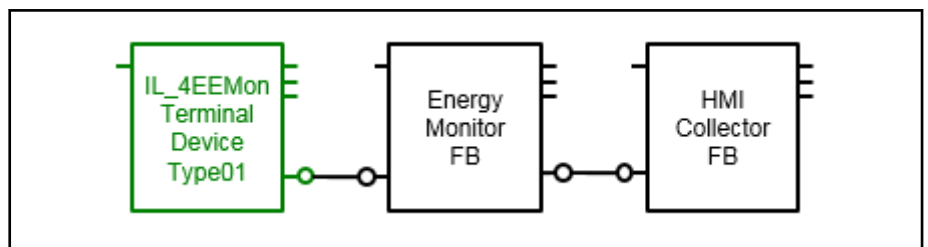


Fig. 9-17: Integrating IL_4EEMonTerminalDeviceType01 into EnergyMonitoring

The blocks are connected via interfaces. In order to link the blocks, the instance of the EnergyMonitor block with which the instance of the IL_4EEMonTerminalDeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonTerminalDevice: IL_4EEMonTerminalDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR
```

Maximum Number of Blocks Registering

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

The number of device blocks that can be registered at an EnergyControl block can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in [Parameter list, page 297](#). The default value is 84.

Configuration

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEHMIDataCollectorType01 are divided into

RIL_4EEEnergyManagement.library

drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

Connect the cyclic input data of the measuring terminals (IN0 - IN11) to the InProcData input. The mode of the measuring terminals is specified via the OutProcData output which has to be the first word (OUT0) to be connected to the cyclic output data of the energy measuring terminal. The bit-coded diagnostics of the measuring terminals are output at the TerminalDiag output. The structure is contained in the following table:

Bit	Description	Note
0	Current input phase 1 overridden	
1	Current input phase 2 overridden	
2	Current input phase 3 overridden	
3	Current input neutral conductor overridden	
4	Voltage input phase 1 overridden	
5	Voltage input phase 2 overridden	
6	Voltage input phase 3 overridden	
7	-	
8	No frequency synchronization	
9	Frequency < 20 Hz	
10	Frequency > 100 Hz	
11	-	
12	Direction of rotation incorrect	LED E, red light on
13	No direction of rotation found	LED E, red light on
14	Phase failure	LED E, red light on
15	-	
16	Polarity reversal Phase 1	Negative effective power
17	Capacitive Phase 1	Negative fundamental wave reactive power
18	Current too low Phase 1	< 10 % of maximum current, < 8% of nominal value
19	-	
20	Polarity reversal Phase 2	Negative effective power
21	Capacitive Phase 2	Negative fundamental wave reactive power
22	Current too low Phase 2	< 10 % of maximum current, < 8% of nominal value
23	-	
24	Polarity reversal Phase 3	Negative effective power
25	Capacitive Phase 3	Negative fundamental wave reactive power

RIL_4EEEnergyManagement.library

26	Current too low Phase 3	< 10 % of maximum current, < 8% of nominal value
27	-	
28	-	
29	-	
30	-	
31	Incorrect direction of rotation	Phases exchanged, transformer connection twisted, only for connection types three-phase current

Tab. 9-34: Structure of the TerminalDiag output

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be less than 32
DEVICE_ERROR	16#2A03	16#0000	Cause: Terminal not calibrated Solution: Terminal has to be serviced
DEVICE_ERROR	16#2A03	16#0001	Cause: Runtime error of firmware solution: A quick bus cycle (< 2 ms), high network frequency (> 80 Hz) and a minimum averaging time can result in overload.
DEVICE_ERROR	16#2A03	16#0002	Cause: Bus cycle too fast in case of a 3-phase synchronous mode (< 4 ms) Solution: Increase bus cycle time
DEVICE_ERROR	16#2A03	16#0003	Cause: Transformer factors invalid Solution: Check the transformer factors for voltage and current in the configuration
DEVICE_ERROR	16#2A03	16#0004	Cause: Configuration invalid Solution: Verify the configuration settings

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
DEVICE_ERROR	16#2A03	16#0005	Cause: Measuring circuit defective
OTHER_ERROR	16#2A02	16#0000	Cause: The block could not be registered with the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR	16#2A02	16#0001	Cause: The device block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one device block can be registered as the TotalConsumption device
OTHER_ERROR	16#2A02	16#0006	Cause: Timeout when changing the mode of the measuring terminal. Either the cyclic data of the measuring terminal are not available at the function block or the measuring terminal does not respond Solution: Check the connection of the measuring terminal to the function block terminal restart
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-35: Error codes IL_4EEMonTerminalDeviceType01 function block

IL_4EEMonTerminalDeviceType02

Brief description

The device function block IL_4EEMonTerminalDeviceType02 is used to read out the inline energy measurement terminal IB IL PM 3P/N/EF-2MBD-PAC - 2700576 via the bus couplers for Sercos (IL S3 BK DI8 DO4 2TX-PAC - 2692380) and Profinet (IL PN BK DI8 DO4 2TX-PAC - 2703994) of Phoenix Contact. The function block is registered at an EnergyMonitor function block to which it transmits the energy data. In addition to the IL_4EEMonTerminalDeviceType01 function block, the IL_4EEMonTerminalDeviceType02 function block can read out other measured objects from the terminal and output these objects.

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-36: Assignment target system/library

Interface description

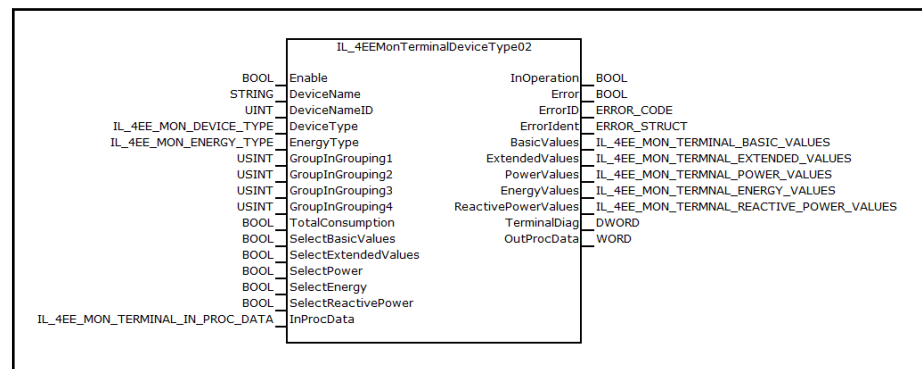


Fig. 9-18: IL_4EEMonTerminalDeviceType02 function block

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	DeviceName	STRING(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name, used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Select device type, used for HMI display See IL_4EE_MON_DEVICE_TYPE, page 194
	EnergyType	IL_4EE_MON_ENERGY_TYPE	Select energy type, used for HMI display See IL_4EE_MON_ENERGY_TYPE, page 194
	GroupInGrouping1	USINT	Group in Grouping 1, used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2, used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3, used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4, used for HMI display
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	SelectBasicValues	BOOL	Enables reading out of phase-to-phase voltage, currents and active powers
	SelectExtendedValues	BOOL	Enables reading out of the phase conductors, the bimetal current measurement and the frequency
	SelectPowerValues	BOOL	Enables reading out of the reactive power, the apparent power and the power factors
	SelectEnergyValues	BOOL	Enables reading out of active energy and reactive energy and the operating hour counter
	SelectReactivePowerValues	BOOL	Enables reading out of active energy and reactive energy and of cos (phi)
	InProcData	IL_4EE_MON_TERMINAL_IN_PROC_DATA	Cyclic input data of the energy measurement terminal See IL_4EE_MON_TERMINAL_IN_PROC_DATA, page 196
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Troubleshooting, page 147

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	BasicValues	IL_4EE_MON_TERMINAL_BASIC_VALUES	Measured values of voltage between phases, currents and active power See IL_4EE_MON_TERMINAL_BASIC_VALUES , page 196
	ExtendedValues	IL_4EE_MON_TERMINAL_EXTENDED_VALUES	Measured values of the phase conductors, the bimetal current measurements and the frequency See IL_4EE_MON_TERMINAL_EXTENDED_VALUES , page 196
	PowerValues	IL_4EE_MON_TERMINAL_POWER_VALUES	Measured values of the reactive power, the apparent power and the power factors See IL_4EE_MON_TERMINAL_POWER_VALUES , page 197
	EnergyValues	IL_4EE_MON_TERMINAL_ENERGY_VALUES	Measured values of the active energy and the reactive energy as well as the operating hour counter See IL_4EE_MON_TERMINAL_ENERGY_VALUES , page 198
	ReactivePowerValues	IL_4EE_MON_TERMINAL_REACTIVE_POWER_VALUES	Measured values of the reactive power and of cos (phi) See IL_4EE_MON_TERMINAL_REACTIVE_POWER_VALUES , page 198
	TerminalDiag	DWORD	Bit-coded output of terminal diagnostics
	OutProcData	WORD	Cyclic output data of the energy measurement terminal contains the control word to select the measuring object

Tab. 9-37: IL_4EEMonTerminalDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. Edge Enable
DeviceNameID	USINT			0	Pos. Edge Enable
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. Edge Enable
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. Edge Enable
GroupInGrouping1	USINT	0	63	0	Pos. Edge Enable
GroupInGrouping2	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping3	USINT	0	31	0	Pos. Edge Enable
GroupInGrouping4	USINT	0	31	0	Pos. Edge Enable
TotalConsumption	BOOL			FALSE	Pos. Edge Enable
SelectBasicValues	BOOL			FALSE	Pos. Edge Enable

Name	Type	Min. value	Max. value	Default value	Effective
SelectExtended-Values	BOOL			FALSE	Pos. Edge Enable
SelectPowerValues	BOOL			FALSE	Pos. Edge Enable
SelectEnergyValues	BOOL			FALSE	Pos. Edge Enable
SelectReactivePowerValues	BOOL			FALSE	Pos. Edge Enable
InProcData	IL_4EE_MON_TERMINAL_IN_PRO C_DATA				Continuous

Tab. 9-38: Min./max. and default values of the IL_4EEMonTerminalDeviceType02 inputs

Functional description

The IL_4EEMonTerminalDeviceType02 function block registers at a central EnergyMonitor function block during instantiation.

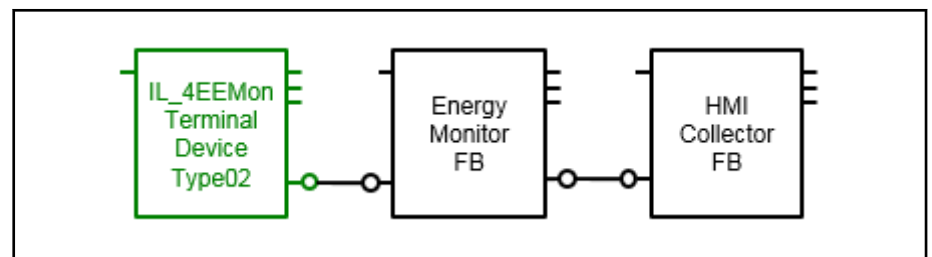


Fig. 9-19: Integrating IL_4EEMonTerminalDeviceType02 into Energy Monitoring

The blocks are connected via interfaces. In order to link the function blocks, the instance of the EnergyMonitor block at which the instance of the IL_4EEMonTerminalDeviceType02 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonTerminalDevice: IL_4EEMonTerminalDeviceType02(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR
```

Maximum Number of Blocks Registering

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

The number of device blocks that can be registered at an EnergyControl block can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in [Parameter list, page 297](#),. The default value is 84.

Configuration

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

RIL_4EEEnergyManagement.library

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

Via the inputs, the measured objects SelectBasicValues, SelectExtendedValues, SelectPowerValues, SelectEnergyValues und SelectReactivePowerValues can be selected from the terminal. The selected measured objects are subsequently read out from the terminal (multiplex) via the process data. The objects are then output at the respected output. This means, the more measured objects are selected, the greater the update time for each individual structure.

Connect the cyclic input data of the measuring terminals (IN0 - IN11) to the InProcData input. The mode of the measuring terminals is specified via the OutProcData output which has to be the first word (OUT0) to be connected to the cyclic output data of the energy measuring terminal. The bit-coded diagnostics of the measuring terminals are output at the TerminalDiag output. The structure is contained in the following table:

Bit	Description	Note
0	Current input phase 1 overridden	
1	Current input phase 2 overridden	
2	Current input phase 3 overridden	
3	Current input neutral conductor overridden	
4	Voltage input phase 1 overridden	
5	Voltage input phase 2 overridden	
6	Voltage input phase 3 overridden	
7	-	
8	No frequency synchronization	
9	Frequency < 20 Hz	
10	Frequency > 100 Hz	
11	-	
12	Direction of rotation incorrect	LED E, red light on
13	No direction of rotation found	LED E, red light on
14	Phase failure	LED E, red light on
15	-	
16	Polarity reversal Phase 1	Negative effective power
17	Capacitive Phase 1	Negative fundamental wave reactive power
18	Current too low Phase 1	< 10 % of maximum current, < 8% of nominal value
19	-	

RIL_4EEEnergyManagement.library

20	Polarity reversal Phase 2	Negative effective power
21	Capacitive Phase 2	Negative fundamental wave reactive power
22	Current too low Phase 2	< 10 % of maximum current, < 8% of nominal value
23	-	
24	Polarity reversal Phase 3	Negative effective power
25	Capacitive Phase 3	Negative fundamental wave reactive power
26	Current too low Phase 3	< 10 % of maximum current, < 8% of nominal value
27	-	
28	-	
29	-	
30	-	
31	Incorrect direction of rotation	Phases exchanged, transformer connection twisted, only for connection types three-phase current

Tab. 9-39: Structure of the TerminalDiag output

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be less than 64
INPUT_RANGE_ERROR	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be less than 32
INPUT_RANGE_ERROR	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be less than 32
DEVICE_ERROR	16#2A03	16#0000	Cause: Terminal not calibrated Solution: Terminal has to be serviced
DEVICE_ERROR	16#2A03	16#0001	Cause: Runtime error of firmware solution: A quick bus cycle (< 2 ms), high network frequency (> 80 Hz) and a minimum averaging time can result in overload.

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
DEVICE_ERROR	16#2A03	16#0002	Cause: Bus cycle too fast in case of a 3-phase synchronous mode (< 4 ms) Solution: Increase bus cycle time
DEVICE_ERROR	16#2A03	16#0003	Cause: Transformer factors invalid Solution: Check the transformer factors for voltage and current in the configuration
DEVICE_ERROR	16#2A03	16#0004	Cause: Configuration invalid Solution: Verify the configuration settings
DEVICE_ERROR	16#2A03	16#0005	Cause: Measuring circuit defective
OTHER_ERROR	16#2A02	16#0000	Cause: The block could not be registered with the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR	16#2A02	16#0001	Cause: The device block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one device block can be registered as the TotalConsumption device
OTHER_ERROR	16#2A02	16#0006	Cause: Timeout when changing the mode of the measuring terminal. Either the cyclic data of the measuring terminal are not available at the function block or the measuring terminal does not respond Solution: Check the connection of the measuring terminal to the function block terminal restart
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-40: Error codes of the IL_4EEMonTerminalDeviceType02 function block

IL_4EEMonCompAirDeviceType01

Brief description

The "Device" function block IL_4EEMonCompAirDeviceType01 is used to generically gather energy and power data for consumers of compressed air. The function block is registered in an EnergyMonitor function block. The function block determines the energy consumption of compressed air using measured values. To determine the energy, the pressure unit and the flow rate unit can be set at the function block.

The function block provides the following methods as an interface:

- mSetGasFlowMeter
- mSetPressure

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-41: Assignment: Target system/library

Interface description

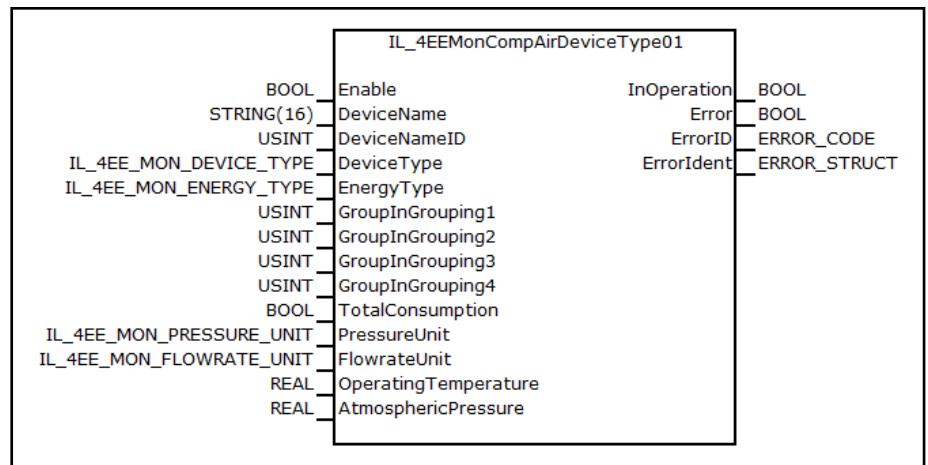


Fig. 9-20: IL_4EEMonCompAirDeviceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	DeviceName	STRING(16)	Name of device, used for HMI display
	DeviceName ID	USINT	ID of the device name; used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Selection of device type; used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	EnergyType	IL_4EE_MON_ENERGY_TYPE	Selection of energy type; used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1; used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2; used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3; used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4; used for HMI display
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	FlowRateUnit	IL_4EE_MON_FLOWRATE_UNIT	Selection of flow rate unit See IL_4EE_MON_FLOWRATE_UNIT(ENUM) , page 199
	PressureUnit	IL_4EE_MON_PRESSURE_UNIT	Selection of pressure unit See IL_4EE_MON_PRESSURE_UNIT , page 199
	AtmosphericPressure	REAL	Ambient pressure in bar
	OperatingTemperature	REAL	Operating temperature in °C
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostics, see Troubleshooting, page 152

Tab. 9-42: IL_4EEMonCompAirDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. edge "Enable"
DeviceNameID	USINT			0	Pos. edge "Enable"
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. edge "Enable"
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. edge "Enable"
GroupInGrouping 1	USINT	0	63	0	Pos. edge "Enable"
GroupInGrouping 2	USINT	0	31	0	Pos. edge "Enable"
GroupInGrouping 3	USINT	0	31	0	Pos. edge "Enable"
GroupInGrouping 4	USINT	0	31	0	Pos. edge "Enable"
TotalConsumption	BOOL			FALSE	Pos. edge "Enable"
FlowRateUnit	IL_4EE_MON_FLOWRATE_UNIT			FLOWRATE_L_PER_SECOND	Pos. edge "Enable"
PressureUnit	IL_4EE_MON_PRESSURE_UNIT			PRESSURE_UNIT_BAR	Pos. edge "Enable"
AtmosphericPressure	REAL	0		1.01325	Pos. edge "Enable"
OperationTemperature	REAL	-273.15		20	Pos. edge "Enable"

Tab. 9-43: Min./max. and default values of the IL_4EEMonCompAirDeviceType01 inputs

Function block methods

The function block provides the following methods as an interface:

- [mSetGasFlowMeter, page 150](#)
- [mSetPressure, page 151](#)

mSetGasFlowMeter

The "mSetGasFlowMeter" method is used to set the current value of the flow meter for the consumer in the function block. This method is only relevant if the function block is in the InOperation state.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	StandardAirFlowRate	REAL			Standard flow rate of the consumer
VAR_INPUT	StandardAirConsumedQuantity	REAL			Current value of the flow meter

Tab. 9-44: Interface of the "mSetGasFlowMeter" method with min./max. values



The method "mSetGasFlowMeter" may only write on one input (either StandardAirFlowRate or StandardAirConsumedQuantity). The second input has to be 0.

mSetPressure

The "mSetPressure" method is used to set the current pressure of the gas for the consumer in the function block. This method is only relevant if the function block is in the InOperation state.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	PressureValue	REAL			Current flow rate meter pressure

Tab. 9-45: Interface of the "mSetPressure" method with min./max. values

Functional description

The IL_4EECompAirDeviceType01 function block registers on a central EnergyMonitor block during instantiation.

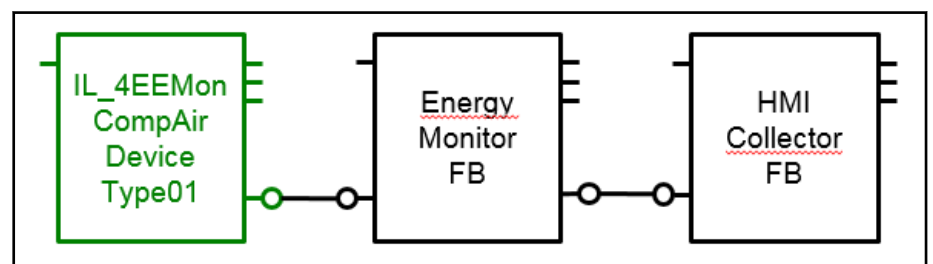


Fig. 9-21: Integrating IL_4EEMonCompAirDeviceType01 into energy monitoring

The blocks are connected via interfaces. In order to link the function blocks, the instance of the EnergyMonitor block at which the instance of the IL_4EEMonGasDeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonCompAirDevice: IL_4EEMonCompAirDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR
```

RIL_4EEEnergyManagement.library

	The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified at program runtime.
Maximum number of blocks to be registered	The number of device blocks, that can be registered in an EnergyControl block, can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in Parameter list, page 297 ,. The default value is 84.
Configuration	A flow rate unit can be specified by the input "FlowRateUnit" via an ENUM. The pressure unit can be specified by the "PressureUnit" input via an ENUM. The inputs "AtmosphericPressure" and "OperatingTemperature" set the ambient pressure and the operating temperature. The specification of the current gas flow rate or the consumed air is given via the inputs "StandardFlowRate" or "StandardAirConsumedQuantity" of the "mSetGasFlowMeter" method. The method may only write on one of the two variables. The second one has to be 0. The TotalConsumption input can configure the block as the total power consumption counter for the machine. The following inputs divide the block into various groups: • DeviceType: Determines drives, heaters, units, etc. • EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc. • Grouping1-4: Allows user-defined division into application-specific groups, such as machine module The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE). In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately. Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.
Troubleshooting	The function block uses the error table F_RELATED_TABLE, 16#0170 . It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be lower than 32

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000C	Cause: Input "AtmosphericPressure" <= 0 Solution: The input "AtmosphericPressure" has to be greater than 0.
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000D	Cause: Input "OperationTemperature" < -273.0 °C Solution: The input OperationTemperature has to be greater than -273.0 °C.
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#001C	Cause: Both method inputs of the method "mSetGasFlowMeter" of the function block are called and unequal to 0. Solution: Only one method input of the function block may be unequal to 0 at a time (either "StandardAirFlowRate" or "StandardAirConsumedQuantity")
OTHER_ERROR (16#7FFE)	16#2A02	16#0000	Cause: The block could not be registered in the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR (16#7FFE)	16#2A02	16#0001	Cause: The "Device" function block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one "Device" block can be registered as the TotalConsumption device
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-46: Error codes of the IL_4EEMonCompAirDeviceType01 function block

IL_4EEMonFluidDeviceType01

Brief description The device function block IL_4EEMonFluidDeviceType01 is used to generically gather energy and power data for consumers of fluids and gases where energy is transmitted via temperature differences (e.g. cooling water, heating gas). The function block is registered in an EnergyMonitor function block. The function block determines the energy consumption of the medium using the flow rate. To determine the energy, the medium parameters (type, heat capacity and density), the temperature of the medium and the unit of the volume flow rate can be set at the function block. If no matching medium type can be selected, a customized type with the respective parameters can be set.

The function block provides the following methods as an interface:

- mSetFlowMeter

RIL_4EEEnergyManagement.library

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-47: Assignment: Target system/library

Interface description

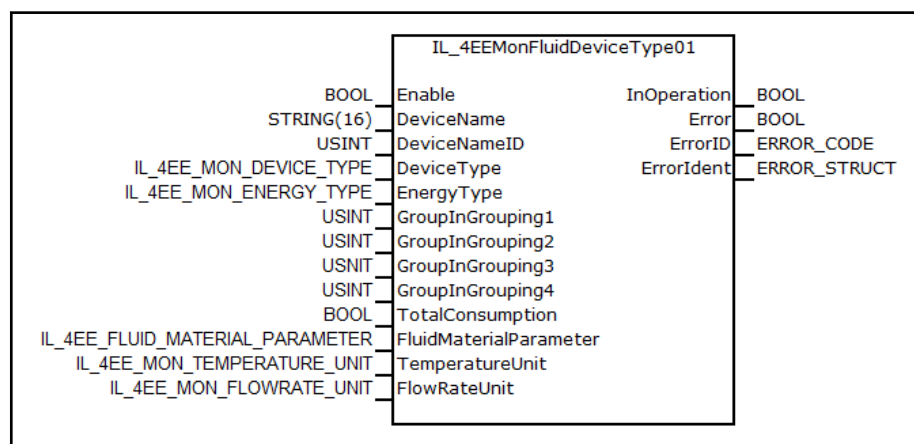


Fig. 9-22: IL_4EEMonFluidDeviceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	Device-Name	STRING(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name; used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Selection of device type; used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	Energy-Type	IL_4EE_MON_ENERGY_TYPE	Selection of energy type; used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1; used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2; used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3; used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4; used for HMI display
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	FluidMaterialParameter	IL_4EE_FLUID_MATERIAL_PARAMETER	Selection of the parameter for the medium type, density and heat capacity See IL_4EE_FLUID_MATERIAL_PARAMETER , page 199
	TemperatureUnit	IL_4EE_MON_TEMPERATURE_UNIT	Specification of the temperature unit See IL_4EE_MON_TEMPERATURE_UNIT(ENUM) , page 200

I/O type	Name	Data type	Description
	FlowRateUnit	IL_4EE_MON_FLOWRATE_UNIT	Selection of flow rate unit See IL_4EE_MON_FLOWRATE_UNIT(ENUM) , page 199
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics, see Troubleshooting , page 157

Tab. 9-48: IL_4EEMonFluidDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. edge "Enable"
DeviceNameID	USINT			0	Pos. edge "Enable"
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. edge "Enable"
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. edge "Enable"
GroupInGrouping1	USINT	0	63	0	Pos. edge "Enable"
GroupInGrouping2	USINT	0	31	0	Pos. edge "Enable"
GroupInGrouping3	USINT	0	31	0	Pos. edge "Enable"
GroupInGrouping4	USINT	0	31	0	Pos. edge "Enable"
TotalConsumption	BOOL			FALSE	Pos. edge "Enable"
FluidMaterial-Parameter	IL_4EE_FLUID_MATERIAL_PARAMETER				Pos. edge "Enable"
TemperaturUnit	IL_4EE_MON_TEMPERATURE_UNIT			TEMPERATURE_UNIT_CELSIUS	Pos. edge "Enable"
FlowRateUnit	IL_4EE_MON_FLOWRATE_UNIT			FLOW-RATE_M3_PER_HOUR	Pos. edge "Enable"

Tab. 9-49: Min./max. and default values of the IL_4EEMonFluidDeviceType01 inputs

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Description
FluidMaterialType	IL_4EE_MON_FLUID_MATERIAL_TYPE			FLUID_TYPE_WATER	
FluidDensity	REAL	0.1	20000.0	998.2	Water at 25 °C [kg/m³]
FluidHeatCapacity	REAL	100.0	20000.0	4181.30	Water at 25 °C [J/(kg*K)]

Tab. 9-50: Min./max. and default values of the IL_4EE_FLUID_MATERIAL_PARAMETER data structure

Function block methods The function block provides the following methods as an interface:

- [mSetFlowMeter](#), page 156

mSetFlowMeter The “mSetFlowMeter” method sets the current value of the flow meter and the temperature in the function block. This method is only relevant if the function block is in the InOperation state.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no “FlowRate” and “Temperature” values applied
VAR_INPUT	FlowRate	REAL			Current value of the flow meter Remark: If the flow meter provides volumes instead of flow rate values, they have to be converted to a flow rate (average flow rate in the last call interval of this method)
VAR_INPUT	Temperature	REAL			Current temperature value

Tab. 9-51: Interface of the “mSetFlowMeter” method with min./max. values

Functional description The IL_4EEMonFluidDeviceType01 function block registers on a central EnergyMonitor block during instantiation.

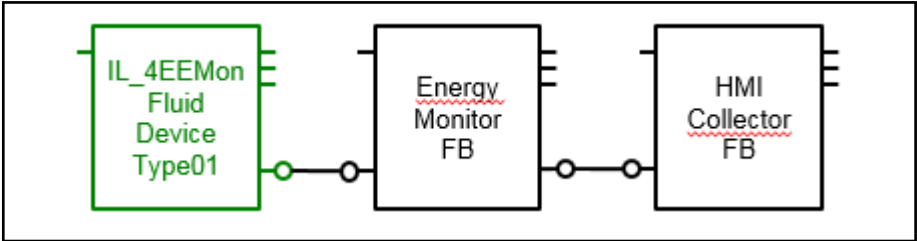


Fig. 9-23: Integrating IL_4EEMonFluidDeviceType01 into energy monitoring

The blocks are connected via interfaces. In order to link the function blocks, the instance of the EnergyMonitor block at which the instance of the IL_4EE-MonFluidDeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

Program:

```

VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonFluidDevice: IL_4EEMonFluidDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR
  
```

Maximum number of blocks to be registered

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified at program runtime.

The number of device blocks, that can be registered in an EnergyControl block, can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in [Parameter list, page 297](#),. The default value is 84.

Configuration

The FluidType can be preselected by the “FluidMaterialParameter” input via an ENUM and the respective parameters, e.g. density and heat capacity, can be specified. If the matching FluidType cannot be used, the value FLUID_TYPE_USER_DEFINED can be used. The values for density and heat capacity created by the user are applied in this case.

The input “TemperaturUnit” specifies the temperature unit of the measured value.

The “FlowRateUnit” input specifies the unit to transfer data to the function block (e.g. m³/h). The function block is rescaled in the function block using this unit.

The current gas flow rate and the current temperature are specified via the mSetFlowMeter method. When changing the flow rate/temperature, this value is changed by calling the method again.

The TotalConsumption input can configure the block as the total power consumption counter for the machine.

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0018	Cause: Density < Minimum (FluidDensity) Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0019	Cause: Density > Maximum (FluidDensity) Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#001A	Cause: Heat capacity < Minimum (FluidHeatCapacity) Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#001B	Cause: Heat capacity > Maximum (FluidHeatCapacity) Solution: Adjust value
OTHER_ERROR (16#7FFE)	16#2A02	16#0000	Cause: The block could not be registered in the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR (16#7FFE)	16#2A02	16#0001	Cause: The "Device" function block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one "Device" block can be registered as the TotalConsumption device
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-52: Error codes of the IL_4EEMonFluidDeviceType01 function block

IL_4EEMonGasDeviceType01

Brief description The function block IL_4EEMonGasDeviceType01 is used to generically gather energy and power data for consumers of combustible gases. The function block is registered in an "EnergyMonitor" function block. The function block

RIL_4EEEnergyManagement.library

determines the energy consumption of a combustion gas using measured values. To determine the energy, the gas type (calorific value) and the flow rate unit can be set at the function block. If no matching gas type can be selected, a customized gas type can be set with the respective calorific value (unit in kWh/m³).

The function block provides the following methods as an interface:

- mSetFlowMeter

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-53: Assignment: Target system/library

Interface description

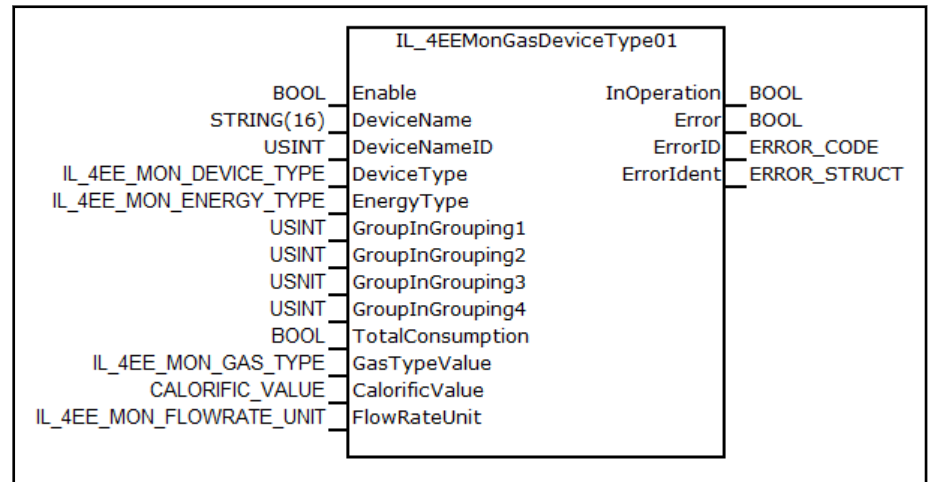


Fig. 9-24: IL_4EEMonGasDeviceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	Device-Name	STRING(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name; used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Selection of device type; used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	Energy-Type	IL_4EE_MON_ENERGY_TYPE	Selection of energy type; used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1; used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2; used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3; used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4; used for HMI display

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	GasTypeValue	IL_4EE_MON_GAS_TYPE	Selection for the "GasType"; used to determine the calorific value See IL_4EE_MON_GAS_TYPE(ENUM) , page 200
	CalorificValue	REAL	Specification of a calorific value (unit kWh/m³). Only valid if GasTypeValue = GAS_TYPE_USER_DEFINED
	FlowRateUnit	IL_4EE_MON_FLOWRATE_UNIT	Selection of flow rate unit See IL_4EE_MON_FLOWRATE_UNIT(ENUM) , page 199
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics, see Troubleshooting , page 162

Tab. 9-54: IL_4EEMonGasDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. edge "Enable"
DeviceNameID	USINT			0	Pos. edge "Enable"
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. edge "Enable"
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. edge "Enable"
GroupInGrouping1	USINT	0	63	0	Pos. edge "Enable"
GroupInGrouping2	USINT	0	31	0	Pos. edge "Enable"
GroupInGrouping3	USINT	0	31	0	Pos. edge "Enable"
GroupInGrouping4	USINT	0	31	0	Pos. edge "Enable"
TotalConsumption	BOOL			FALSE	Pos. edge "Enable"
GasTypeValue	IL_4EE_MON_GAS_TYPE			GAS_TYPE_NATURAL_GAS	Pos. edge "Enable"
CalorificValue	REAL	0	100	10	Pos. edge "Enable"
FlowRateUnit	IL_4EE_MON_FLOWRATE_UNIT			FLOW-RATE_M3_PER_HOUR	Pos. edge "Enable"

Tab. 9-55: Min./max. and default values of the IL_4EEMonGasDeviceType01 inputs

Function block methods

The function block provides the following methods as an interface:

- [mSetFlowMeter](#), page 161

mSetFlowMeter The “mSetFlowMeter” method sets the current value of the flow meter for the consumer in the function block. This method is only relevant if the function block is in the InOperation state.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method could not be executed successfully and does not apply any values of the “FlowMeterValue”
VAR_INPUT	FlowMeterValue	REAL			Current value of the flow meter. Remark: If the flow meter provides volumes instead of flow rate values, they have to be converted to a flow rate (average flow rate in the last call interval of this method)

Tab. 9-56: Interface of the “mSetFlowMeter” method with min./max. values

Functional description The IL_4EEMonGasDeviceType01 function block registers on a central EnergyMonitor block during instantiation.

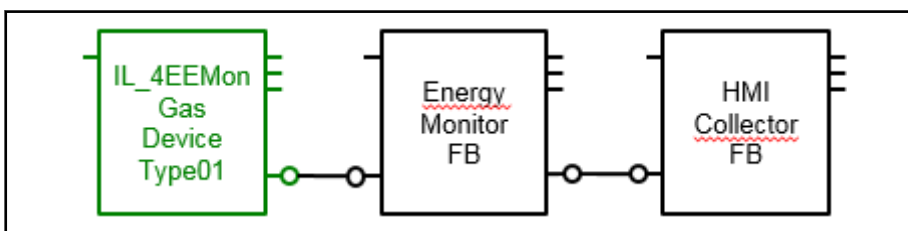


Fig. 9-25: Integrating IL_4EEMonGasDeviceType01 into energy monitoring

The blocks are connected via interfaces. In order to link the function blocks, the instance of the EnergyMonitor block at which the instance of the IL_4EEMonGasDeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```

VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonGasDevice: IL_4EEMonGasDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);;
END_VAR
  
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. Registering cannot be modified at program runtime.

Maximum number of function blocks to be registered

The number of device blocks, that can be registered in an EnergyControl block, can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in [Parameter list, page 297](#),. The default value is 84.

Configuration

A GasType can be preselected by the “GasType” input via an ENUM. Depending on the gas type set, the respective calorific value is set in the function block. If the matching gas type cannot be selected, the value USER_DEFINED_GAS_TYPE can be used. In this case, the respective calorific value is specified in kWh/m³ at the “CalorificValue” input.

RIL_4EEEnergyManagement.library

The “CalorificValue” input can be used to send a customized calorific value (unit kWh/m³) to the function block. The value is only evaluated if the value USER_DEFINED_GAS_TYPE is applied at the “GasType” input.

The “FlowRateUnit” input specifies the unit to transfer data to the function block (e.g. m³/h). The function block is rescaled in the function block using this unit.

The current gas flow rate is specified via the “mSetFlowMeter” method. When changing the flow rate, this value is changed by calling the method again.

The TotalConsumption input can configure the block as the total power consumption counter for the machine.

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be lower than 32

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000A	Cause: The input CalorificValue is not within the valid range ($0 < \text{CalorificValue} \leq 100$) Solution: The input "CalorificValue" has to remain within the limits
OTHER_ERROR (16#7FFE)	16#2A02	16#0000	Cause: The block could not be registered in the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR (16#7FFE)	16#2A02	16#0001	Cause: The "Device" function block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one "Device" block can be registered as the TotalConsumption device
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-57: Error codes of the IL_4EEMonGasDeviceType01 function block

IL_4EEMonGasDeviceType02

Brief description

The "Device" function block IL_4EEMonGasDeviceType02 is used to generically gather energy and power data for consumers of combustible gases. The function block is registered in an "EnergyMonitor" function block. The function block determines the energy consumption of a combustion gas using measured values. To determine the energy, the gas type (calorific value) can be set at the function block. If no matching gas type can be selected, a customized gas type can be set with the respective calorific value (unit in kWh/m³). The S0 pulses are sent to the function block via an "mSetS0Signal()" method.

The function block provides the following methods as an interface:

- mSetS0Signal

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-58: Assignment: Target system/library

RIL_4EEEnergyManagement.library

Interface description

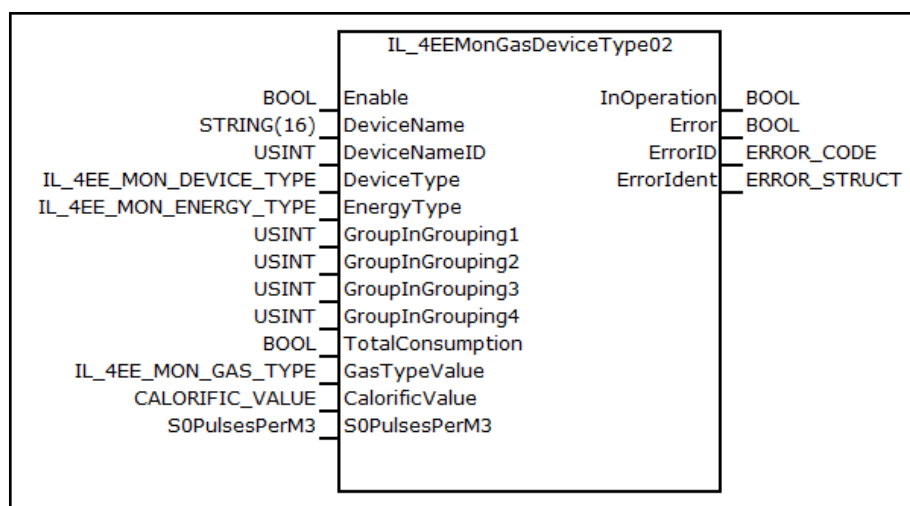


Fig. 9-26: IL_4EEMonGasDeviceType02 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	Device-Name	STRING(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name; used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Selection of device type; used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	Energy-Type	IL_4EE_MON_ENERGY_TYPE	Selection of energy type; used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1; used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2; used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3; used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4; used for HMI display
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	GasTypeValue	IL_4EE_MON_GAS_TYPE	Selection for the "GasType"; used to determine the calorific value See IL_4EE_MON_GAS_TYPE(ENUM) , page 200
	Calorific-Value	REAL	Specification of a calorific value (unit kWh/m³). Only valid if GasTypeValue = GAS_TYPE_USER_DEFINED
	S0PulsesPerM3	UINT	Number of S0 pulses per m³
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error

I/O type	Name	Data type	Description
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics, see Troubleshooting, page 167

Tab. 9-59: IL_4EEMonGasDeviceType02 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. edge "Enable"
DeviceNameID	USINT			0	Pos. edge "Enable"
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. edge "Enable"
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. edge "Enable"
GroupInGrouping1	USINT	0	63	0	Pos. edge "Enable"
GroupInGrouping2	USINT	0	31	0	Pos. edge "Enable"
GroupInGrouping3	USINT	0	31	0	Pos. edge "Enable"
GroupInGrouping4	USINT	0	31	0	Pos. edge "Enable"
TotalConsumption	BOOL			FALSE	Pos. edge "Enable"
GasTypeValue	IL_4EE_MON_GAS_TYPE			GAS_TYPE_NATURAL_GAS	Pos. edge "Enable"
CalorificValue	REAL	0	100	10	Pos. edge "Enable"
S0PulsesPerM3	UINT	1		100	Pos. edge "Enable"

Tab. 9-60: Min./max. and default values of the IL_4EEMonGasDeviceType02 inputs

Function block methods The function block provides the following methods as an interface:

- [mSetS0Signal, page 165](#)

mSetS0Signal The method "mSetS0Signal ()" sends S0 pulses to the function block. This method is only relevant if the function block is in the InOperation state.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method could not be executed successfully and does not apply any values of the "FlowMeterValue"
VAR_INPUT	S0Signal	REAL			S0 pulses

Tab. 9-61: Interface of the "mSetFlowMeter" method with min./max. values

Functional description The IL_4EEMonGasDeviceType01 function block registers on a central EnergyMonitor block during instantiation.

RIL_4EEEnergyManagement.library

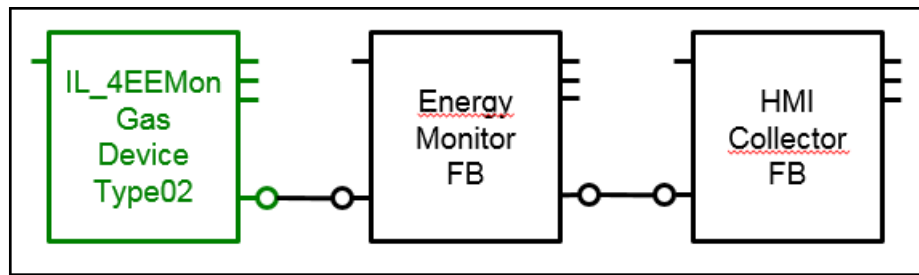


Fig. 9-27: Integrating IL_4EEMonGasDeviceType02 into energy monitoring

The blocks are connected via interfaces. In order to link the function blocks, the instance of the EnergyMonitor block at which the instance of the IL_4EE-MonGasDeviceType02 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

Program:

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonGasDevice: IL_4EEMonGasDeviceType02(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified at program runtime.

Maximum number of function blocks to be registered

The number of device blocks, that can be registered in an EnergyControl block, can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in [Parameter list, page 297](#),. The default value is 84.

Configuration

A gas type can be preselected by the "GasType" input via an ENUM. Depending on the gas type set, the respective calorific value is set in the function block. If the matching gas type cannot be selected, the value USER_DEFINED_GAS_TYPE can be used. In this case, the respective calorific value is specified in kWh/m³ at the "CalorificValue" input.

The "CalorificValue" input can be used to send a customized calorific value (unit kWh/m³) to the function block. The value is only evaluated if the value USER_DEFINED_GAS_TYPE is applied at the "GasType" input.

The TotalConsumption input can configure the block as the total power consumption counter for the machine.

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

The method "mSetS0Signal(S0Signal=)" provides the pulses of the flow meter to the function block.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000A	Cause: The input CalorificValue is not within the valid range ($0 < \text{CalorificValue} \leq 100$) Solution: The input "CalorificValue" has to remain within the limits
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000B	Cause: Value for "S0PulsesPerkWh" is 0 Solution: The input value has to be greater than 0
OTHER_ERROR (16#7FFE)	16#2A02	16#0000	Cause: The block could not be registered in the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR (16#7FFE)	16#2A02	16#0001	Cause: The "Device" function block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one "Device" block can be registered as the TotalConsumption device

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
OTHER_ERROR (16#7FFE)	16#2A02	16#0009	Cause: The S0 pulses do not correspond to the specification (high pulses not in the range 30 - 120 ms / low pulses < 30 ms) Solution: The S0 pulses have to meet the specification
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-62: Error codes of the IL_4EEMonGasDeviceType02 function block

IL_4EEMonPneumaticDeviceType01

Brief description The "Device" function block IL_4EEMonPneumaticDeviceType01 is used to generically gather energy and power data of pneumatic cylinders. The function block is registered in an EnergyMonitor function block. The function block determines the energy consumption of a pneumatic cylinder using measured values. To determine the energy, the parameters for the cylinder can be set at the function block. Furthermore, there is an input for the ambient temperature and the ambient pressure. The function block outputs the air consumption in standard cubic meters at one output.

The function block provides the following methods as an interface:

- mSetCylinderPosition
- mSetPressure

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-63: Assignment: Target system/library

Interface description

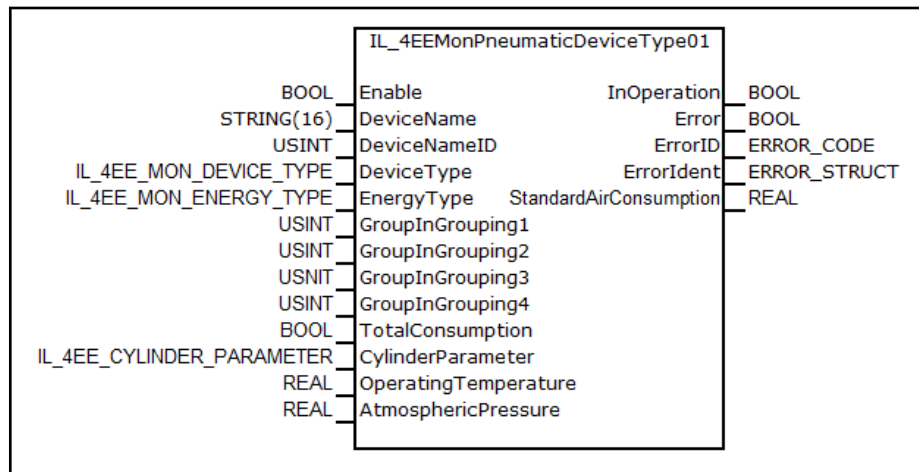


Fig. 9-28: IL_4EEMonPneumaticDeviceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	Device-Name	STRING(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name; used for display on HMIS with language switching

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Selection of device type; used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	Energy-Type	IL_4EE_MON_ENERGY_TYPE	Selection of energy type; used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1; used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2; used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3; used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4; used for HMI display
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	Cylinder-Parameter	IL_4EE_CYLINDER_PARAMETER	Technical parameters of the cylinder, incl. connections See IL_4EE_CYLINDER_PARAMETER , page 200
	AtmosphericPressure	REAL	Ambient pressure in bar
	Operating-Temperature	REAL	Operating temperature in °C
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics, see Troubleshooting , page 173
	StandardAirConsumption	REAL	Air consumption in standard cubic meters

Tab. 9-64: IL_4EEMonPneumaticDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. edge "Enable"
DeviceNameID	USINT			0	Pos. edge "Enable"
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. edge "Enable"
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. edge "Enable"
GroupInGrouping1	USINT	0	63	0	Pos. edge "Enable"

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective
GroupInGroup- ing2	USINT	0	31	0	Pos. edge "Enable"
GroupInGroup- ing3	USINT	0	31	0	Pos. edge "Enable"
GroupInGroup- ing4	USINT	0	31	0	Pos. edge "Enable"
TotalConsumption	BOOL			FALSE	Pos. edge "Enable"
CylinderParame- ter	IL_4EE_CYLIN- DER_PARAME- TER				Pos. edge "Enable"
AtmosphericPres- sure	REAL	0		1.01325	Pos. edge "Enable"
OperatingTem- perature	REAL	-273.15		20	Pos. edge "Enable"

Tab. 9-65: Min./max. and default values of the IL_4EEMonPneumaticDevice-Type01 inputs

Min./max. and default values of IL_4EE_CYLINDER_PARAMETER

Name	Type	Min. value	Max. value	Default	Description
CylinderPistonDia- meter	REAL	0		32	Piston diameter 32 mm
CylinderPistonRod- DiameterExtension	REAL	0		0	Cylinder with one piston rod
CylinderPistonRod- DiameterReturn	REAL	0		12	Diameter of the piston rod in direction of retraction is 12 mm
StrokeLength	REAL	0		500	Stroke length 500 mm
ExtensionHoseDia- meter	REAL	0		4	Connection diameter at extension site 4 mm
ReturnHoseDiamete- r	REAL	0		4	Connection diameter at retraction site 4 mm
ExtensionHose- Length	REAL	0		1000	Connection length at extension site 1000 mm
ReturnHoseLength	REAL	0		2000	Connection length at retraction site 2000 mm

Tab. 9-66: Min./max., default of IL_4EE_CYLINDER_PARAMETER

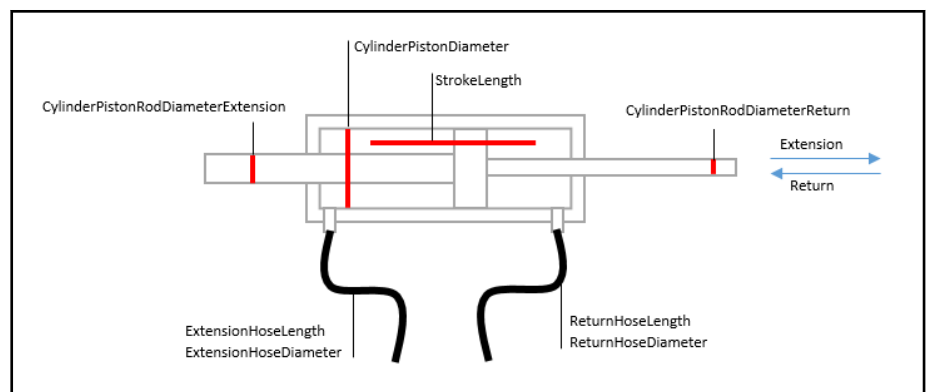


Fig. 9-29: CylinderParameter

Function block methods

The function block provides the following methods as an interface:

- - mSetPressure, page 171
- - mSetCylinderPosition, page 171

- mSetPressure

The “mSetPressure” method sets the current pressure of the flow meter for the consumer in the function block. This method is only relevant if the function block is in the InOperation state.

I/O type	Name	Type	Description
RETURN_VALUE		BOOL	TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	PressureValueExtension	REAL	Current value of the pressure sensor when extending, unit in bar
VAR_INPUT	PressureValueReturn	REAL	Current value of the pressure sensor when retracting, unit in bar

Tab. 9-67: Interface of the mSetFlowMeter method

- mSetCylinderPosition

The method “mSetCylinderPosition” sets the current position of the cylinder. This method is only relevant if the function block is in the InOperation state.

I/O type	Name	Type	Description
RETURN_VALUE		BOOL	TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	CylinderPosition	BOOL	Current cylinder position TRUE: Extended, in end position FALSE: Retracted, in end position

Tab. 9-68: Interface of the mSetCylinderPosition method

Functional description

The IL_4EEMonPneumaticDeviceType01 function block registers on a central EnergyMonitor block during instantiation.

RIL_4EEEnergyManagement.library

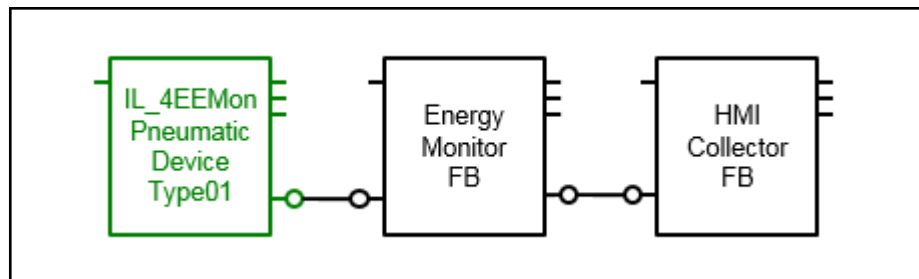


Fig. 9-30: Integrating IL_4EEMonPneumaticDeviceType01 into energy monitoring

The blocks are connected via interfaces. In order to link the function blocks, the instance of the EnergyMonitor block at which the instance of the IL_4EEMonPneumaticDeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonPneumaticDevice: IL_4EEMonPneumaticDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);;
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified at program runtime.

Maximum number of function blocks to be registered

The number of device blocks, that can be registered in an EnergyControl block, can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in [Parameter list, page 297](#). The default value is 84.

Configuration

The “CylinderParameter” structure can be used to enter the parameters required to write on the cylinder (refer to the figure).

The operating temperature is specified via the “OperatingTemperature” input.

The input “AtmosphericPressure” represents the ambient pressure.

The TotalConsumption input can configure the block as the total power consumption counter for the machine.

The following inputs divide the block into various groups:

- DeviceType: Determines drives, heaters, units, etc.
- EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc.
- Grouping1-4: Allows user-defined division into application-specific groups, such as machine module

The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE).

In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately.

Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.

Troubleshooting The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000C	Cause: Input "AtmosphericPressure" <= 0 Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000D	Cause: Value OperatingTemperature <-273.0 °C Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000E	Cause: Value CylinderPistonDiameter < 0 Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000F	Cause: Value CylinderPistonRodDiameterExtension < 0 Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0010	Cause: Value CylinderPistonRodDiameterReturn < 0 Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0011	Cause: Value StrokeLength < 0 Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0012	Cause: Value ExtensionHoseDiameter < 0 Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0013	Cause: Value ReturnHoseDiameter < 0 Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0014	Cause: Value ExtensionHoseLength < 0 Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0015	Cause: Value ReturnHoseLength < 0 Solution: Adjust value

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0016	Cause: Value of the method "mSetPressureValueExtension" < 0 Solution: Adjust value
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0017	Cause: Value of the method "mSetPressureValueReturn" < 0 Solution: Adjust value
OTHER_ERROR (16#7FFE)	16#2A02	16#0000	Cause: The block could not be registered in the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR (16#7FFE)	16#2A02	16#0001	Cause: The "Device" function block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one "Device" block can be registered as the TotalConsumption device
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-69: Error codes of the IL_4EEMonPneumaticDeviceType01 function block

IL_4EEMonS0DeviceType01

Brief description The "Device" function block IL_4EEMonS0DeviceType01 analyzes energy consumption values transmitted via a pulse interface acc. to EN 62053-31 (S0 interface). The function block is registered at an EnergyMonitor function block to which it transmits the energy data.

The function block provides the following method as an interface:

- mSetS0Signal

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-70: Assignment: Target system/library

Interface description

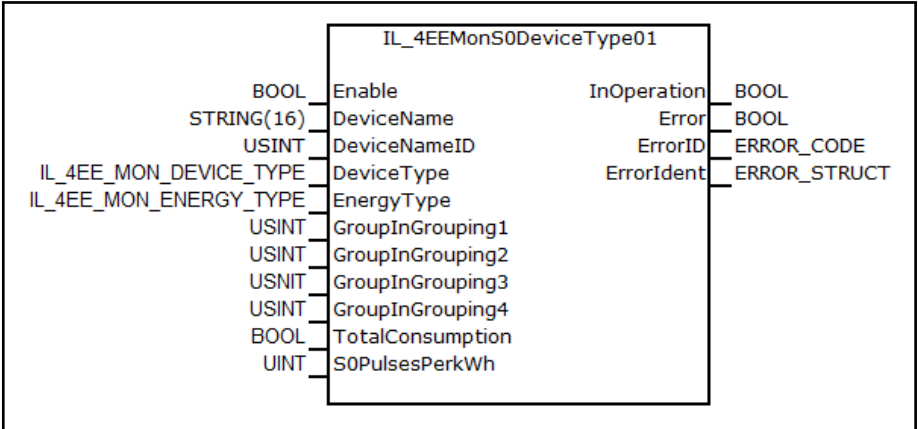


Fig. 9-31: IL_4EEMonS0DeviceType01 function block

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	Device-Name	STRING(16)	Name of device, used for HMI display
	DeviceNameID	USINT	ID of the device name; used for display on HMIS with language switching
	DeviceType	IL_4EE_MON_DEVICE_TYPE	Selection of device type; used for HMI display See IL_4EE_MON_DEVICE_TYPE , page 194
	Energy-Type	IL_4EE_MON_ENERGY_TYPE	Selection of energy type; used for HMI display See IL_4EE_MON_ENERGY_TYPE , page 194
	GroupInGrouping1	USINT	Group in Grouping 1; used for HMI display
	GroupInGrouping2	USINT	Group in Grouping 2; used for HMI display
	GroupInGrouping3	USINT	Group in Grouping 3; used for HMI display
	GroupInGrouping4	USINT	Group in Grouping 4; used for HMI display
	TotalConsumption	BOOL	TRUE - this instance is the total energy counter, only possible with one instance per EnergyMonitor block
	S0PulsesPerkWh	UINT	Number of S0 pulses per kWh
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics, see Troubleshooting , page 177

Tab. 9-71: IL_4EEMonS0DeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
DeviceName	STRING(16)			"	Pos. edge "Enable"
DeviceNameID	USINT			0	Pos. edge "Enable"
DeviceType	IL_4EE_MON_DEVICE_TYPE	DEV_TYPE_UNDEFINED	DEV_TYPE_USER_DEFINED_63	DEV_TYPE_UNIT	Pos. edge "Enable"
EnergyType	IL_4EE_MON_ENERGY_TYPE	E_TYPE_UNDEFINED	E_TYPE_USER_DEFINED_31	E_TYPE_ELECTRICAL	Pos. edge "Enable"
GroupInGrouping1	USINT	0	63	0	Pos. edge "Enable"
GroupInGrouping2	USINT	0	31	0	Pos. edge "Enable"

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective
GroupInGroup- ing3	USINT	0	31	0	Pos. edge "Enable"
GroupInGroup- ing4	USINT	0	31	0	Pos. edge "Enable"
TotalConsumption	BOOL			FALSE	Pos. edge "Enable"
S0PulsesPerkWh	UINT	1		1000	Pos. edge "Enable"

Tab. 9-72: Min./max. and default values of the IL_4EEMonS0DeviceType01 inputs

Function block methods The function block provides the following methods as an interface:

- [mSetS0Signal, page 176](#)

mSetS0Signal The method "mSetS0Signal" sends S0 pulses to the function block. This method is only relevant if the function block is in the InOperation state.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no values of S0 pulses applied
VAR_INPUT	S0Signal	BOOL			S0 pulses of energy meter

Tab. 9-73: Interface of the "mSetS0Signal" method with min./max. values

Functional description The IL_4EEMonS0DeviceType01 function block registers on a central EnergyMonitor block during instantiation.

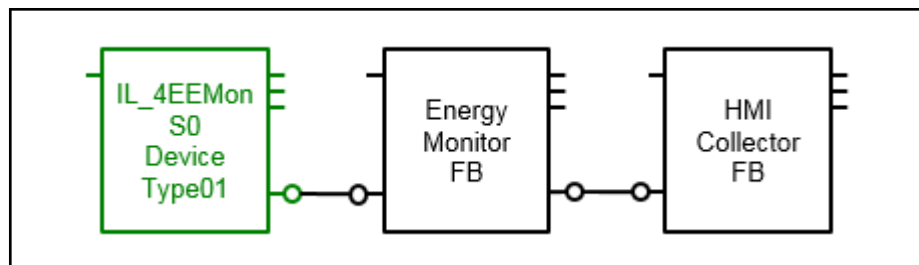


Fig. 9-32: Integrating IL_4EEMonS0DeviceType01 into energy monitoring

The blocks are connected via interfaces. To link the function blocks, the instance of the EnergyMonitor block at which the instance of the IL_4EEMonS0DeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the consumption values.

SYNTAX

Program:

```

VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonS0Device: IL_4EEMonS0DeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);
END_VAR

```

Maximum number of function blocks to be registered	The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified at program runtime.
	The number of device blocks, that can be registered in an EnergyControl block, can be modified with the constant NO_OF_MON_DEVICES_PER_ENERGY_MONITOR in Parameter list, page 297 ,. The default value is 84.
Configuration	The number of pulses output by the sensor when consuming 1 kWh is sent to the function block via the "S0PulsesPerkWh". S0 pulses are transferred via the method "mSetS0Signal". The TotalConsumption input can configure the block as the total power consumption counter for the machine. The following inputs divide the block into various groups: • DeviceType: Determines drives, heaters, units, etc. • EnergyType: Determines form of energy, such as electrical energy, mechanical energy, etc. • Grouping1-4: Allows user-defined division into application-specific groups, such as machine module The group divisions can be used for the HMI display. This means, e.g., the consumers for block IL_4EEMonHMIDataCollectorType01 are divided into drives (DeviceType = DEV_TYPE_DRIVE) and units (DeviceType ≠ DEV_TYPE_DRIVE). In the same manner, using the EnergyType input can show the consumption of electrical energy and mechanical or thermal energy separately. Using the freely configurable Grouping1-4 inputs can create application-specific groupings. For example, the input Grouping1 can assign the consumers to machine modules.
Troubleshooting	The function block uses the error table F_RELATED_TABLE, 16#0170 . It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be lower than 32

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR (16#0006)	16#2A00	16#000B	Cause: Value for "S0PulsesPerkWh" is 0 Solution: The input value has to be greater than 0
OTHER_ERROR (16#7FFE)	16#2A02	16#0009	Cause: The S0 pulses do not correspond to the specification (high pulses not in the range 30 - 120 ms / low pulses < 30 ms) Solution: The S0 pulses have to meet the specification
OTHER_ERROR (16#7FFE)	16#2A02	16#0000	Cause: The block could not be registered in the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR (16#7FFE)	16#2A02	16#0001	Cause: The "Device" function block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one "Device" block can be registered as the TotalConsumption device
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-74: Error codes of the IL_4EEMonS0DeviceType01 function block

IL_4EELoadProfileEstimation

Brief description

The function block IL_4EELoadProfileEstimation creates a load profile estimation on the incoming energy consumption values. The function block refers to an IL_4EEEnergyMonitorType01 and the consumed energy of one or multiple devices saved in the odometer. By extrapolating the respective current energy value to a definable interval length, the expected interval load profile can be estimated. The estimation is inhibited at the beginning of the interval, as the estimation values are still very volatile and become only stable enough after a certain period (ca. 20% of the interval) to work with them. There are two energy limit values (lower/upper limit) as input and limit value violations of the estimation as output. The total energy consumption of the measurement and the energy consumption of the last interval can be retrieved at the output.

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-75: Assignment: Target system/library

Interface description

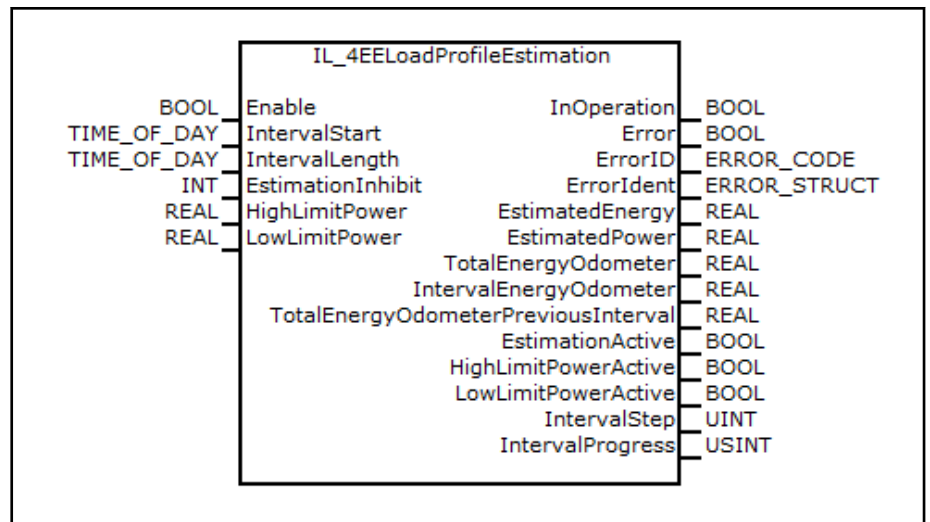


Fig. 9-33: IL_4EELoadProfileEstimation function block

I/O type	Name	Data type	Description
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	EstimationInhibit	USINT	Period at the beginning of an interval with inhibited estimation. Indicated in percent [%]
	IntervalStart	TIME_OF_DAY	Time specifying the interval start
	IntervalLength	TIME	Interval length
	HighLimitPower	REAL	Upper limit value of medium power in the interval in [W]
	LowLimitPower	REAL	Lower limit value of medium power in the power in [W]
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics, see troubleshooting
	EstimatedEnergy	REAL	Estimated total energy consumption of the current interval in [Ws]
	EstimatedPower	REAL	Estimated average power of the current interval in [W]
	TotalEnergyOdometer	REAL	Total energy consumption (odometer) since function block enabling in [Ws]
	IntervalEnergyOdometer	REAL	Energy consumption (odometer) in the current interval in [Ws]

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	TotalEnergyOdometerPreviousInterval	REAL	Total energy consumption (odometer) of the previous interval [Ws]
	EstimationActive	BOOL	Estimation active if TRUE
	HighLimitPowerActive	BOOL	Upper limit value exceeded if TRUE
	LowLimitPowerActive	BOOL	Lower limit value exceeded if TRUE
	IntervalStep	UINT	This output is incremented upon every started estimation interval. The counter is provided with an overflow at 65535 and starts again at 0
	IntervalProgress	USINT	Progress of the current interval in [%]

Tab. 9-76: Interface description of the IL_4EELoadProfileEstimation function block

The energy values (EstimatedEnergy, TotalEnergyOdometer, IntervalEnergyOdometer, TotalEnergyOdometerPreviousInterval) can assume positive and negative values. Positive values mean energy consumption, negative values mean energy output.

Min./max. and default values and applying the inputs

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A00	16#0000	Cause: Value of DeviceType input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR	16#2A00	16#0001	Cause: Value of EnergyType input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR	16#2A00	16#0002	Cause: Value of GroupInGroupings1 input outside permitted range Solution: The value of the input must be lower than 64
INPUT_RANGE_ERROR	16#2A00	16#0003	Cause: Value of GroupInGroupings2 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR	16#2A00	16#0004	Cause: Value of GroupInGroupings3 input outside permitted range Solution: The value of the input must be lower than 32
INPUT_RANGE_ERROR	16#2A00	16#0005	Cause: Value of GroupInGroupings4 input outside permitted range Solution: The value of the input must be lower than 32
DEVICE_ERROR	16#2A03	16#0000	Cause: Terminal not calibrated Solution: Terminal has to be serviced

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
DEVICE_ERROR	16#2A03	16#0001	Cause: Runtime error of the firmware Solution: A quick bus cycle (< 2 ms), high network frequency (> 80 Hz) and a minimum averaging time can result in overload
DEVICE_ERROR	16#2A03	16#0002	Cause: Bus cycle too fast in case of a 3-phase synchronous mode (< 4 ms) Solution: Increase bus cycle time
DEVICE_ERROR	16#2A03	16#0003	Cause: Transformer factors invalid Solution: Check the transformer factors for voltage and current in the configuration
DEVICE_ERROR	16#2A03	16#0004	Cause: Configuration invalid Solution: Verify the configuration settings
DEVICE_ERROR	16#2A03	16#0005	Cause: Measuring circuit defective
OTHER_ERROR	16#2A02	16#0000	Cause: The block could not be registered in the EnergyMonitor block as a device block Solution: The NO_OF_MON_DEVICES_PER_ENERGY_MONITOR constant has to be increased in the parameter list
OTHER_ERROR	16#2A02	16#0001	Cause: The "Device" function block could not be registered with the EnergyMonitor block as the TotalConsumption device Solution: Only one "Device" block can be registered as the TotalConsumption device
OTHER_ERROR	16#2A02	16#0006	Cause: Timeout when changing the mode of the measuring terminal. Either the cyclic data of the measuring terminal are not available at the function block or the measuring terminal does not respond Solution: Check the connection of the measuring terminal to the function block terminal restart
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-77: Error codes of the IL_4EELoadProfileEstimation function block

Functional description

It is referenced to an "EnergyMonitoring" function block of the IL_4EEEnergyMonitorType01 via an interface of the type "ITF_EnergyMonitor". Upon each call of the function block "IL_4EELoadProfileEstimation" via this interface, the current value of the "EnergyMonitoring" function block (stTotalConsumption.Odometer) is retrieved (cf. figure 5). This value is the sum of the energy consumption of all devices coupled to the "EnergyMonitoring" function block. To estimate load profiles for individual devices, an "EnergyMonitor" function block and a "LoadProfileEstimation" function block are required for each device.

RIL_4EEEnergyManagement.library

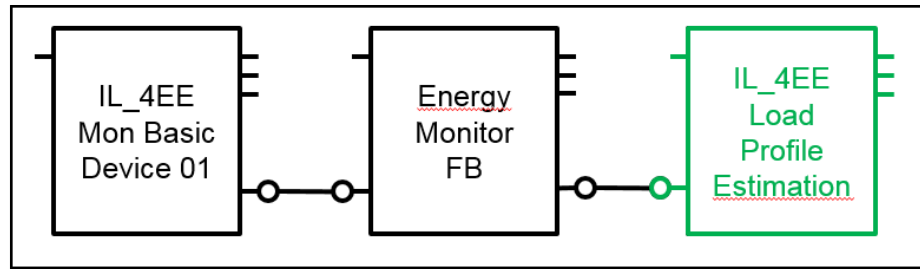


Fig. 9-34: Coupling an IL_4EELoadProfileEstimation function block to an "EnergyMonitor" function block

The function block calculates the estimation of the energy consumption within the interval of the length "IntervalLength" based on a linear extrapolation of the current energy consumption "IntervalEnergyOdometer" in the interval.

If "IntervalStart" is set to TOD#08:04:00 and "IntervalLength" is set T#5M, the first interval is from 08:04:00 o'clock to 08:09:00 o'clock, the second interval is from 08:09:00 o'clock to 08:14:00 o'clock, etc. The additional reference to a current system time allows a synchronization between the intervals specified by the energy suppliers and the intervals computed in the function block and the load profile determined by the energy suppliers can be understood analogously. Even if the current system time < IntervalStart, the intervals are computed correctly.

A use case of the load profile estimation is shown in the following figure. In a 15 minute interval, the energy consumption expected of the remaining interval is extrapolated. When exceeding the upper limit HighLimitPower, the variable HighLimitPowerActive is set to TRUE. When exceeding the lower limit LowLimitPower, the variable LowLimitPowerActive is set to TRUE. This allows to derive measures in order not to exceed the energy consumption in the load profile interval, e.g. by connecting or disconnecting consumers.

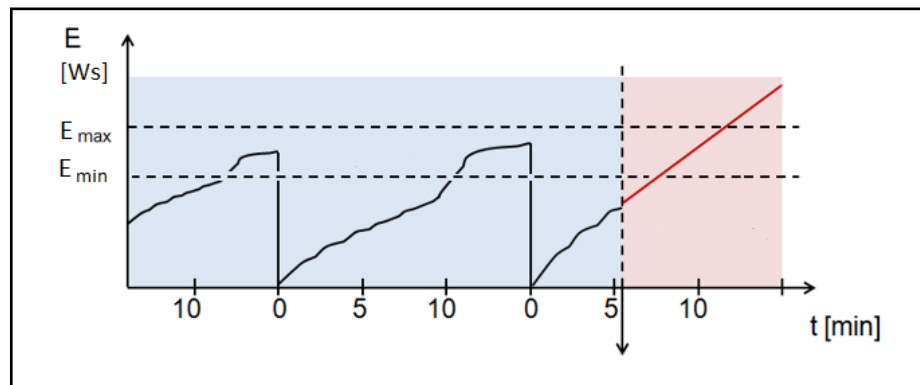


Fig. 9-35:

The "EstimationInhibit" input is used to determine the estimated values only after some time, as the estimation is very inaccurate at the beginning of the interval - especially in case of a continuously changing energy consumption.

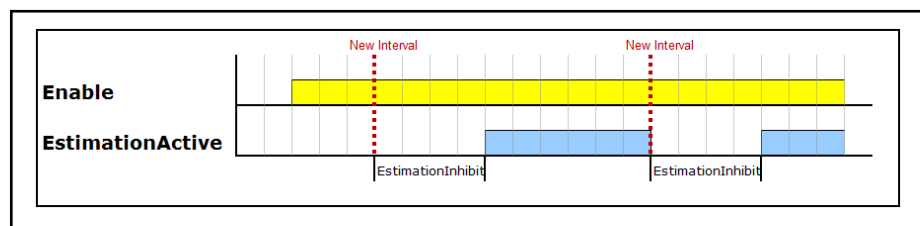


Fig. 9-36: Response of EstimationActive

The “IntervalStep” output is incremented when a new interval begins. At the same time, the “TotalEnergyOdometerPreviousInterval” output is set with the last value of the “IntervalEnergyOdometer” output. Thus, the number of completed intervals can be retrieved directly and subsequent calculations can be made, e.g. the average interval load can be calculated using “TotalEnergyOdometer”. The output “IntervalProgress” shows the progress of the current interval in (in percent). This provides a better overview and allows a better understanding of the value characteristics. The output “EstimationActive” is TRUE if the progress of the interval is outside the inhibit section specified by “EstimationInhibit”. Otherwise, “EstimationActive” is FALSE. “EstimationActive” is also FALSE if the function block is enabled and it is FALSE until the first IntervalStart occurs. This impedes an incorrect estimation of the energy value in the first interval.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A00	16#000A	Cause: “EstimationInhibit” limit value violation Solution: The input value has to be within the defined limit values
INPUT_RANGE_ERROR	16#2A00	16#000B	Cause: “IntervalLength” limit value violation Solution: The input value has to be within the defined limit values
INPUT_INVALID_ERROR	16#2A04	16#0000	Cause: HighLimitPower < LowLimitPower Solution: HighLimitPower has to be greater than LowLimitPower
OTHER_ERROR	16#2A02	16#0008	Cause: Calculation with 0 in the denominator Solution: Ensure that the value of the denominator cannot become 0. This is normally ensured by the limit value specification.
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-78: Error codes of the IL_4EELoadProfileEstimation function block

IL_4EEMonHMIDataCollectorType01

Brief description

The IL_4EEMonHMIDataCollectorType01 block serves to implement the raw data from the EnergyMonitor block on the HMI control interface. Energy can also be measured with the help of the block. The block has to be registered on an EnergyMonitor block.

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-79: Assignment: Target system/library

RIL_4EEEnergyManagement.library

Interface description

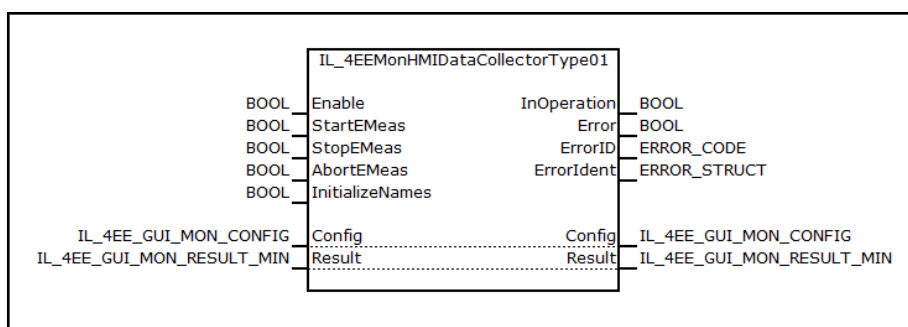


Fig. 9-37: IL_4EEMonHMIDataCollectorType01 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	Config	IL_4EE_GUI_MON_CONFIG	Configuration structure for the HMI, described by IL_4EE_MonHMIDataCollector See IL_4EE_GUI_MON_CONFIG , page 201
	Result	IL_4EE_GUI_MON_RESULT_MIN	Result structure for the HMI, described by IL_4EE_MonHMIDataCollector See IL_4EE_GUI_MON_RESULT_MIN , page 202
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	StartEMeas	BOOL	Starts energy measurement
	StopEMeas	BOOL	Stops energy measurement
	AbortEMeas	BOOL	Aborts energy measurement
	Initialize-Names	BOOL	Reinitializes consumer names in the HMI structure
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostics, see Troubleshooting , page 185

Tab. 9-80: IL_4EEMonHMIDataCollectorType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
StartEMeas	BOOL			FALSE	Continuous
StopEMeas	BOOL			FALSE	Continuous
AbortEMeas	BOOL			FALSE	Continuous
InitializeNames	BOOL			FALSE	Continuous

Tab. 9-81: Minimum/maximum and default values of the IL_4EEMonHMIDataCollectorType01 inputs

Functional description The IL_4EEMonHMIDataCollectorType01 function block registers on a central EnergyMonitor block during instantiation.

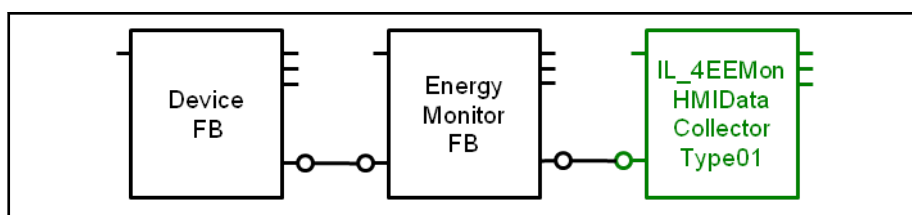


Fig. 9-38: Integrating IL_4EEMonHMIDataCollectorType01 into EnergyMonitoring

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyMonitor block with which the instance of the IL_4EEMonHMIDataCollectorType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block that evaluates the raw data from the EnergyMonitor block and copies it to the HMI structure can be registered.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonBasicDevice: IL_4EEMonBasicDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType01(fbEnergyMonitor);;
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified at program runtime.

Operation

Energy measurements can be started, stopped and aborted with the StartE-Meas, StopE-Meas and AbortE-Meas inputs. If the HMICollector block or the EnergyMonitor block are disabled by the Enable input during measurement, the current measurement is aborted.

Only drives (DEV_TYPE_DRIVE) and units (DEV_TYPE_UNIT) are distinguished in the HMI controls for this block. Therefore, when using these controls, all consumers that do not have the value DEV_TYPE_DRIVE for the DeviceType are shown as units.

With the InitializeNames input, the current names in the Device block inputs are read and copied to the HMI structure. This allows the language to be changed without turning the blocks on or off.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
OTHER_ERROR (16#7FFE)	16#2A02	16#0002	Cause: IL_4EE_NO_OF_DRIVES + IL_4EE_NO_OF_UNITS is greater than NO_OF_MON_DEVICES_PER_ENERGY_MONITOR Solution: Configure NO_OF_MON_DEVICES_PER_ENERGY_CONTROL to be greater than IL_4EE_NO_OF_DRIVES + IL_4EE_NO_OF_UNITS in the parameter list
OTHER_ERROR (16#7FFE)	16#2A02	16#0003	Cause: More drives added than IL_4EE_NO_OF_DRIVES Solution: Increase IL_4EE_NO_OF_DRIVES in the parameter list (max. 64)

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
OTHER_ERROR (16#7FFE)	16#2A02	16#0004	Cause: More "not drives" added than IL_4EE_NO_OF_UNITS Solution: Increase IL_4EE_NO_OF_UNITS in the parameter list (max. 20)
OTHER_ERROR (16#7FFE)	16#2A02	16#0005	Cause: Registration on EnergyMonitor block failed Solution: Specify EnergyMonitor block in the declaration
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-82: Error codes of IL_4EEMonHMIDataCollectorType01 function block

IL_4EEMonHMIDataCollectorType02

Brief description The IL_4EEMonHMIDataCollectorType02 block serves to implement the raw data from the EnergyMonitor block on the web visualization interface. Energy can also be measured with the help of the block. The block has to be registered on an EnergyMonitor block.

In contrast to the IL_4EEMonHMIDataCollectorType03 function block, the function block provides the names of consumers, groups, etc. as strings. Thus, language switching in the HMI is not possible.

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-83: Assignment: Target system/library

Interface description

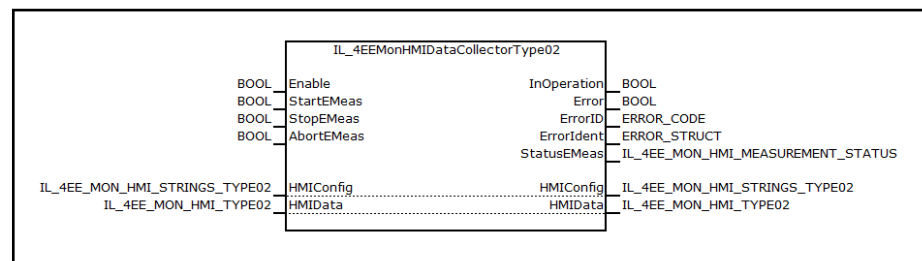


Fig. 9-39: IL_4EEMonHMIDataCollectorType02 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	HMICfg	IL_4EE_MON_HMI_STRINGS_TYPE02	The structure contains the names of measured variables, groupings and groups for the HMI See IL_4EE_MON_HMI_STRINGS_TYPE02, page 207
	HMIDat	IL_4EE_MON_HMI_TYPE02	The HMI interface structure contains the configuration values and measured values for the representation See IL_4EE_MON_HMI_TYPE02, page 206
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	StartMeas	BOOL	Start of the controlled energy measurement
	StopMeas	BOOL	Stop of the controlled energy measurement

I/O type	Name	Data type	Description
	AbortEMeas	BOOL	Cancels the controlled energy measurement
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostics, see Troubleshooting, page 188
	StatusEMeas	IL_4EE_MON_HMI_MEASUREMENT_STATUS	Current energy measurement state See IL_4EE_MON_HMI_MEASUREMENT_STATUS, page 210

Tab. 9-84: IL_4EEMonHMIDataCollectorType02 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
StartEMeas	BOOL			FALSE	Continuous
StopEMeas	BOOL			FALSE	Continuous
AbortEMeas	BOOL			FALSE	Continuous

Tab. 9-85: Min./max. and default values of the IL_4EEMonHMIDataCollectorType02 inputs

Functional description

The IL_4EEMonHMIDataCollectorType02 function block registers on a central EnergyMonitor block during instantiation.

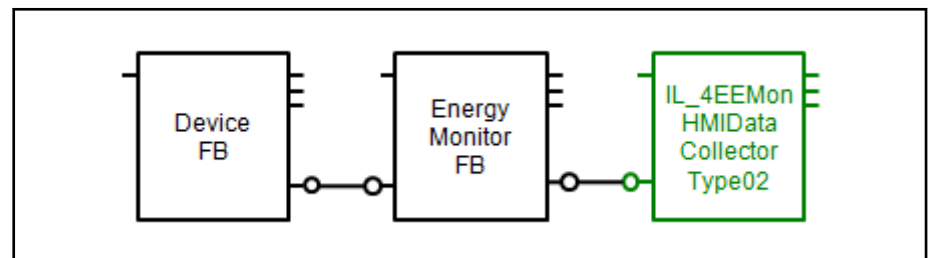


Fig. 9-40: Integrating IL_4EEMonProfiEnergyDeviceType02 into EnergyMonitoring

The blocks are connected via interfaces. In order to link the blocks, the instance of the EnergyMonitor block with which the instance of the IL_4EEMonHMIDataCollectorType02 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block that evaluates the raw data from the EnergyMonitor block and copies it to the HMI structure can be registered.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonBasicDevice: IL_4EEMonBasicDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
```

RIL_4EEEnergyManagement.library

```
fbHMICollector: IL_4EEHMIDataCollectorType02(fbEnergyMonitor);;
END_VAR
```

Operation

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified at program runtime.

Energy measurements can be started, stopped and aborted with the StartE-Meas, StopE-Meas and AbortE-Meas inputs. If the HMICollector block or the EnergyMonitor block are disabled by the Enable input during measurement, the current measurement is aborted. Simultaneously, the same measurement can be started, stopped and canceled via the visualization.

The function block provides the names of devices, groups, etc. as strings in the interface. The names of the consumers have to be specified at the DeviceName input of the device function blocks. All other strings are created at the input/output of HMIConfig of the function block via the configuration structure. By using this function block, language switching is not possible in the visualization.

Troubleshooting

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
OTHER_ERROR (16#7FFE)	16#2A02	16#0005	Cause: Registration on EnergyMonitor block failed Solution: Specify EnergyMonitor block in the declaration
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-86: Error codes of the IL_4EEMonHMIDataCollectorType02 function block

IL_4EEMonHMIDataCollectorType03**Brief description**

The IL_4EEMonHMIDataCollectorType03 block serves to implement the raw data from the EnergyMonitor block on the web visualization interface. Energy can also be measured with the help of the block. The block has to be registered on an EnergyMonitor block.

In contrast to the IL_4EEMonHMIDataCollectorType02 function block, the function block provides the names of consumers, groups, etc. as strings. Thus, language switching in the HMI is possible.

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-87: Assignment: Target system/library

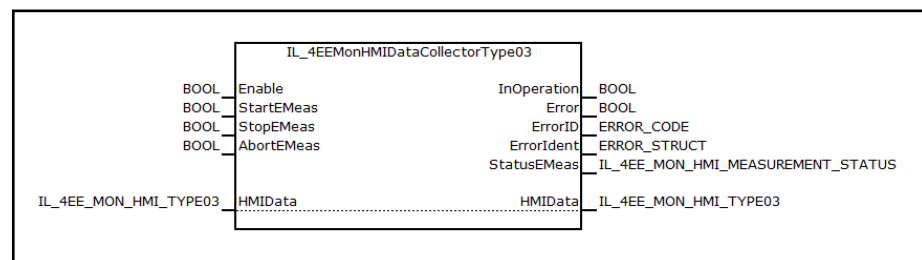
Interface description

Fig. 9-41: IL_4EEMonHMIDataCollectorType03 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	HMIData	IL_4EE_MON_HMI_TYPE03	The HMI interface structure contains the configuration values and measured values for the representation See IL_4EE_MON_HMI_TYPE03 , page 208
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	StartEMeas	BOOL	Start of the controlled energy measurement
	StopEMeas	BOOL	Stop of the controlled energy measurement
	AbortEMeas	BOOL	Cancels the controlled energy measurement
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostics, see Troubleshooting , page 190
	StatusEMeas	IL_4EE_MON_HMI_MEASUREMENT_STATUS	Current energy measurement state See IL_4EE_MON_HMI_MEASUREMENT_STATUS , page 210

Tab. 9-88: IL_4EEMonHMIDataCollectorType03 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
StartEMeas	BOOL			FALSE	Continuous
StopEMeas	BOOL			FALSE	Continuous
AbortEMeas	BOOL			FALSE	Continuous

Tab. 9-89: Min./max. and default values of the IL_4EEMonHMIDataCollectorType03 inputs

Functional description

The IL_4EEMonHMIDataCollectorType03 function block registers on a central EnergyMonitor block during instantiation.

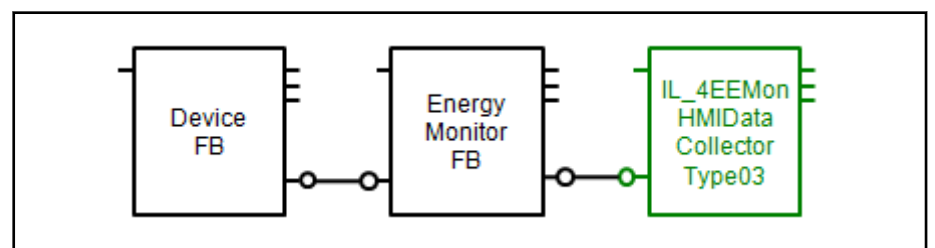


Fig. 9-42: Integrating IL_4EEMonProfiEnergyDeviceType03 into EnergyMonitoring

The blocks are connected via interfaces. In order to link the blocks, the instance of the EnergyMonitor block with which the instance of the IL_4EEMonHMIDataCollectorType03 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block that

RIL_4EEEnergyManagement.library

evaluates the raw data from the EnergyMonitor block and copies it to the HMI structure can be registered.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor-FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a Device-FB, which is registered at the EnergyMonitor-FB *)
fbMonBasicDevice: IL_4EEMonBasicDeviceType01(fbEnergyMonitor);
(* Instance of an HMI-Collector-FB, which is registered at the EnergyMonitor-FB *)
fbHMICollector: IL_4EEHMIDataCollectorType03(fbEnergyMonitor);
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified at program runtime.

Operation Energy measurements can be started, stopped and aborted with the StartE-Meas, StopEMeas and AbortEMeas inputs. If the HMICollector block or the EnergyMonitor block are disabled by the Enable input during measurement, the current measurement is aborted. Simultaneously, the same measurement can be started, stopped and canceled via the visualization.

The function block does not provide any strings as names of devices, groups, etc. in the interface. The strings are read out of a configuration file by the visualization. The names of the consumers have to be assigned at the Device-NameID input of the device function blocks. These Ids correspond to the indexes of the corresponding strings in the DeviceName array in the configuration file. All other strings are also taken from the configuration file. This facilitates an individually adjustable language switching in the visualization.

Troubleshooting The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
OTHER_ERROR (16#7FFE)	16#2A02	16#0005	Cause: Registration on EnergyMonitor block failed Solution: Specify EnergyMonitor block in the declaration
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-90: Error codes of the IL_4EEMonHMIDataCollectorType03 function block

IL_4EEMonScadaDataCollectorType01

Brief description The IL_4EEMonScadaDataCollectorType01 function block serves to connect the EnergyMonitoring on the control to a control system/MES. The function block is registered at a EnergyMonitor function block which represents the machine. The communication with the control system is realized via global structures that have to be applied to the function block. Via the global structures, data of the machine energy counter and the consumer energy counter as well as the energy measurement results are provided.

Target system	Library
MTX from 14V06	RIL_4EEEnergyManagement
MLC from 14V06	RIL_4EEEnergyManagement

Tab. 9-91: Assignment target system/library

RIL_4EEEnergyManagement.library

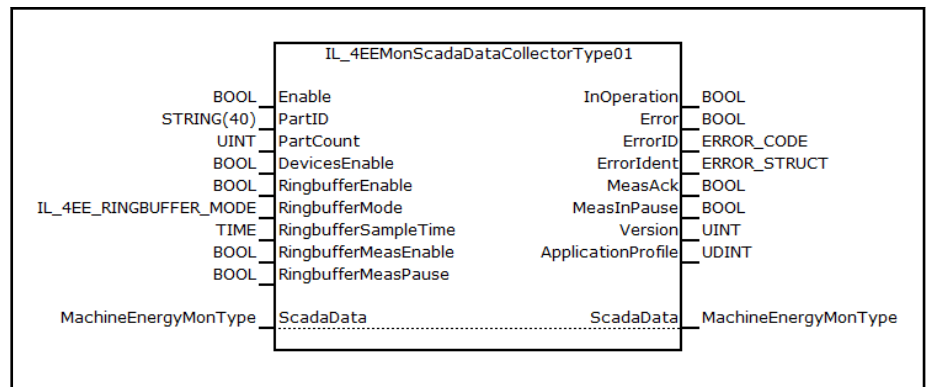


Fig. 9-43: IL_4EEMonScadaDataCollectorType01 function block

Interface description

I/O type	Name	Data type	Description
VAR_IN_OUT	ScadaData	MachineEnergyMonType	Structure with the energy data of the machine and the consumer for communication with the control system/MES MachineEnergyMonType , page 291
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
	PartID	STRING(40)	ID number of the currently manufactured product
	PartCount	UINT	Counter for the manufactured products. A change displays manufacturing of parts. The value can be used by a superordinate instance for e.g. averaging the consumption by using several parts. The value is used internally for the ring buffer management for ring buffers used for parts
	DevicesEnable	BOOL	Enabling the generation of energy consumption data of individual devices (MachineEnergyMonDevices structure)
	RingbufferEnable	BOOL	Enabling the ring buffer of the energy consumption data of the machine (MachineEnergyMonRingbuffer structure)
	Ringbuffer-Mode	IL_4EE_RINGBUFFER_MODE	Measurement mode of the ring buffer. The measurement is actuated either via the RingbufferMeasEnable and RingbufferMeasPause inputs (part-related) or by using the cycle time RingbufferSampleTime (time-related). IL_4EE_RINGBUFFER_MODE , page 296,
	Ringbuffer-SampleTime	TIME	Time-related: Rate during which the ring buffer is filled
	Ringbuffer-MeasEnable	BOOL	Part-related: Start of the next measurement with positive edge. Completing the measurement and writing the measured data in the interface in case of a negative edge
	Ringbuffer-MeasPause	BOOL	Part-related: Pausing the measurement in case of a positive edge and continuing the measurement in case of a negative edge
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Troubleshooting, page 194
	MeasAck	BOOL	Indicates whether a measurement was started for the ring buffer
	MeasInPause	BOOL	Indicates if the current measurement pauses
	Version	UINT	Displays the interface version (0x0001)
	Application-Profile	UDINT	Displays the enabled application profile (is OR-ed with the IL_4EEScada-DataTypes01 display and to be written to the structure)

Tab. 9-92: IL_4EEMonScadaDataCollectorType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
PartID	STRING(40)			"	Continuous
PartCount	UINT			0	Continuous
DevicesEnable	BOOL			FALSE	Pos. Edge at Enable
RingbufferEnable	BOOL			FALSE	Pos. Edge at Enable
Ring buffer mode	IL_4EE_RING_BUFFER_MODE			TIME_RELATED	Pos. Edge at Enable
RingbufferSample-Time	TIME			T#15M	Pos. Edge at Enable
RingbufferMeasEnable	BOOL			FALSE	Continuous
RingbufferMeas-Pause	BOOL			FALSE	Continuous

Tab. 9-93: Min./max. and default values of the inputs

Functional description

The IL_4EEMonScadaDataCollectorType01 function block is used to prepare the EnergyMonitoring data required by the control system/MES. The function block is registered at a EnergyMonitor function block which represents the machine.

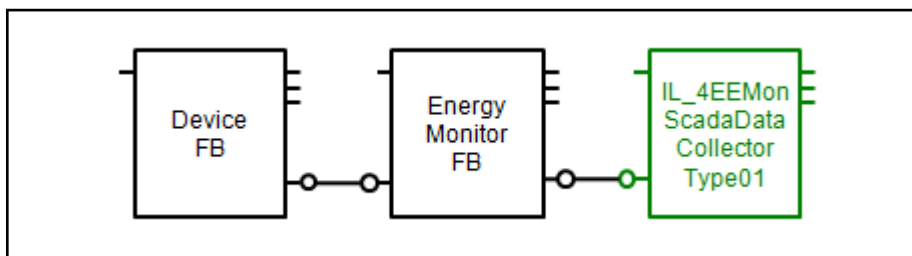


Fig. 9-44: Integrating IL_4EEMonScadaDataCollectorType01 into Energy Management

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyMonitoring block with which the instance of the IL_4EEMonScadaDataCollectorType01 should be registered has to be specified in the declaration as the input of the FB_Init method. The block registers with this link.

SYNTAX

```
VAR
(* Instance of an EnergyMonitor FB *)
fbEnergyMonitor: IL_4EEEnergyMonitorType01;
(* Instance of a device FB which is registered at the EnergyMonitor FB *)
fbMonBasicDevice: IL_4EEMonBasicDeviceType01(fbEnergyMonitor);
(* Instance of an HMI Collector FB which is registered at the EnergyMonitor FB *)
fbScadaDataCollector: IL_4EEScadaDataCollectorType01(fbEnergyMonitor);
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

Call The ScadaDataCollector function block cannot be integrated into the application as other blocks. The block only has to be enabled via the Enable input and called cyclically.

Function The IL_4EEMonScadaDataCollectorType01 function block reads and writes a structure of type MachineEnergyMonType (MachineEnergyObjectType substructure). The structure contains the three substructures:

- MachineEnergyMonConsumption:

The structure contains the energy counter value of the machine of the EnergyMonitor function block. Furthermore, power peaks are displayed as Peak Hold with return movement. Higher performance values are applied directly whereas the display is reduced again (but decelerated) after power peaks occurred. The part-related data PartID, PartDuration and PartCount is always applied by the inputs of the IL_4EEMonScadaDataCollectorType01 function block in case of changes to PartCount. That means, before changing the PartCount input, the manufacturing duration of the last product at the PartDuration input and the ID of the next product have to be applied at the PartID input.

- MachineEnergyMonRingbuffer:

When using the MachineEnergyMonRingbuffer substructure, Ringbuffer-Enable of the function block is enabled by the input. This substructure contains the set BufferMode and the current start index of the ring buffer.

For time-related energy measurements, the measurement interval time of the RingBufferSampleTime of the function block is specified. The measurement data are provided in the next index of the ring buffer after the time interval has expired. The consumed energy of the machine as well as the maximum performance is displayed. For the PartCount value, the increase of the input is displayed during the measurement.

Part-related energy measurement are started level-controlled via the RingbufferMeasEnable input (positive edge) and stopped (negative edge). The measurement data is provided in the next index of the ring buffer after stopping the measurement. The consumed energy of the machine as well as the maximum performance is displayed. The measurement can be interrupted by the RingbufferMeasPause at any time. The energy consumed during the pause is not included in the measured value. The PartDuration value displays the actual measurement time without pauses. The PartID value is applied by the function block input at the end of the measurement. For the PartCount value, the increase of the input is displayed during the measurement.

- MachineEnergyMonDevice:

The consumer data display can be enabled via the DevicesEnable input of the function block. The display indicates the energy counter values as

RIL_4EEEnergyManagement.library

well as the names of the individual machine consumers. The name of the consumers can be updated via a positive edge at the Initialize-Names edge for language switching.

Troubleshooting The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
OTHER_ERROR	16#2A02	16#0005	Cause: Registration on EnergyMonitor block failed Solution: Specify EnergyMonitor block in the declaration
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state
INPUT_RANGE_ERROR	16#00002A00	16#00000009	Cause: Input RingbufferMode is invalid Solution: The input has to have the value TIME_RELATED or PART_RELATED

Tab. 9-94: Error codes of the IL_4EEMonScadaDataCollectorType01 function block

9.2.3 Data types

IL_4EE_MON_DEVICE_TYPE

Brief Description The enumeration is used as the input parameter for device blocks. A specified device type can be selected for each consumer or a user-defined device type can be assigned to the block. The device type can be selected this way when making the selection for display on an HMI.

Element	Value	Description
DEV_TYPE_UNDEFINED	0	No device type defined
DEV_TYPE_DRIVE	1	Devices assigned device type Drives
DEV_TYPE_UNIT	2	Devices assigned device type Units
DEV_TYPE_HEATER	3	Devices assigned device type Heater
DEV_TYPE_HYDRAULIC	4	Devices assigned device type Hydraulic
DEV_TYPE_PNEUMATIC	5	Devices assigned device type Pneumatic
DEV_TYPE_LIGHT	6	Devices assigned device type Lighting
DEV_TYPE_USER_DEFINED_7	7	Device assigned user-defined device type 7
DEV_TYPE_USER_DEFINED_8	8	Device assigned user-defined device type 8
...	...	...
DEV_TYPE_USER_DEFINED_63	63	Device assigned user-defined device type 63

Tab. 9-95: Elements of the IL_4EE_MON_DEVICE_TYPE enumeration type

IL_4EE_MON_ENERGY_TYPE

Brief Description The enumeration is used as the input parameter for device blocks. A specified energy type can be selected for each consumer or a user-defined energy type can be assigned to the block. The energy type can be selected this way when making the selection for display on an HMI.

Element	Value	Description
E_TYPE_UNDEFINED	0	No energy form defined
E_TYPE_ELECTRICAL	1	Consumption of electrical energy
E_TYPE_MECHANICAL	2	Consumption of mechanical energy
E_TYPE_THERMICAL	3	Consumption of thermal energy
E_TYPE_CHEMICAL	4	Consumption of chemical energy
E_TYPE_USER_DEFINED_5	5	Device assigned user-defined device type 5
E_TYPE_USER_DEFINED_6	6	Device assigned user-defined device type 6
...	...	...
E_TYPE_USER_DEFINED_31	31	Device assigned user-defined device type 31

Tab. 9-96: Elements of the IL_4EE_MON_ENERGY_TYPE enumeration type

IL_4EE_MON_PROFI_ENERGY_AVAILABILITY

Brief description The IL_4EE_MON_PROFI_ENERGY_AVAILABILITY enumeration indicates which PROFienergy measured values can be read by the IL_4EEMonProfiEnergyDeviceType01 function block.

Name	Value	Description
NO_MEAS_ID_AVAILABLE	0	No measurement ID found that can use the function block
ACTIVE_ENERGY_SUM_DINT	1	The function block reads out the measurement ID 225 (Active Energy Sum)
ACTIVE_ENERGY_SUM_REAL	2	The function block reads out the measurement ID 205 (Active Energy Sum)
ACTIVE_ENERGY_SUM_LREAL	3	The function block reads out the measurement ID 215 (Active Energy Sum)
ACTIVE_ENERGY_IM_DINT	4	The function block reads out the measurement ID 220 (Active Energy Sum)
ACTIVE_ENERGY_IM_REAL	5	The function block reads out the measurement ID 200 (Active Energy Sum)
ACTIVE_ENERGY_IM_LREAL	6	The function block reads out the measurement ID 210 (Active Energy Sum)
ACTIVE_ENERGY_EX_DINT	7	The function block reads out the measurement ID 221 (Active Energy Sum)
ACTIVE_ENERGY_EX_REAL	8	The function block reads out the measurement ID 201 (Active Energy Sum)
ACTIVE_ENERGY_EX_LREAL	9	The function block reads out the measurement ID 211 (Active Energy Sum)
ACTIVE_ENERGY_IM_EX_DINT	10	The function block reads out the measurement ID 220 (Active Energy Import) and 221 (Active Energy Export)
ACTIVE_ENERGY_IM_EX_REAL	11	The function block reads out the measurement ID 200 (Active Energy Import) and 201 (Active Energy Export)

RIL_4EEEnergyManagement.library

Name	Value	Description
ACTIVE_ENERGY_IM_EX_LREAL	12	The function block reads out the measurement ID 210 (Active Energy Import) and 211 (Active Energy Export)
POWER_INSTANTANEOUS_REAL	13	The function block reads out the measurement ID 34 (Active Power)
POWER_DEMAND_REAL	14	The function block reads out the measurement ID 225 (Active Power)

Tab. 9-97: Elements of the enumeration type IL_4EE_MON_PROFI_ENERGY_AVAILABILITY

IL_4EE_MON_TERMINAL_IN_PROC_DATA

Brief description The structure contains the input raw data of the energy measurement terminal, consisting of the status word and the measured values or diagnostics or the configuration data.

Name	Type	Description
Status	WORD	Status word of the energy measurement terminal
MeasurementValues	ARRAY [1..11] OF WORD	Contains the configuration, the diagnostics or the measured data of the energy measurement terminal, depending on the status word

Tab. 9-98: Design of the IL_4EE_MON_TERMINAL_IN_PROC_DATA structure

IL_4EE_MON_TERMINAL_BASIC_VALUES

Brief description The structure contains the values of voltage between phases, currents and active power measured by the power measurement terminal.

Name	Type	Description
PhaseVoltage1	REAL	Phase voltage 1 [V]
PhaseVoltage2	REAL	Phase voltage 2 [V]
PhaseVoltage3	REAL	Phase voltage 3 [V]
PhaseCurrent1	REAL	Phase current 1 [A]
PhaseCurrent2	REAL	Phase current 2 [A]
PhaseCurrent3	REAL	Phase current 3 [A]
NeutralConductorCurrent	REAL	Neutral conductor current [A]
RealPower1	REAL	Active power 1 [W]
RealPower2	REAL	Active power 2 [W]
RealPower3	REAL	Active power 3 [W]
TotalRealPower	REAL	Active power in total [W]

Tab. 9-99: Design of the IL_4EE_MON_TERMINAL_BASIC_VALUES structure

IL_4EE_MON_TERMINAL_EXTENDED_VALUES

Brief description The structure contains the values of phase conductors, bimetal current measurement and the frequency measured by the power measurement terminal.

Name	Type	Description																														
PhaseConductorVoltage12	REAL	Phase conductor 12 [V]																														
PhaseConductorVoltage23	REAL	Phase conductor 23 [V]																														
PhaseConductorVoltage31	REAL	Phase conductor 31 [V]																														
PhaseCurrent1Filtered	REAL	Phase current 1 filtered [A], time response as for a bimetal measurement mechanism																														
PhaseCurrent2Filtered	REAL	Phase current 2 filtered [A], time response as for a bimetal measurement mechanism																														
PhaseCurrent3Filtered	REAL	Phase current 3 filtered [A], time response as for a bimetal measurement mechanism																														
NeutralConductorCurrentFiltered	REAL	Neutral conductor current filtered [A], time response as for a bimetal measurement mechanism																														
Frequency	REAL	Frequency [Hz]																														
EnergyDirection	WORD	<table> <tr> <td>Bit</td><td>0/1</td><td>=</td></tr> <tr> <td>0</td><td>Consumption/supply</td><td>Phase 1</td></tr> <tr> <td>1</td><td>Consumption/supply</td><td>Phase 2</td></tr> <tr> <td>2</td><td>Consumption/supply</td><td>Phase 3</td></tr> <tr> <td>3</td><td>Consumption/supply</td><td>Total</td></tr> <tr> <td>4</td><td>Inductive/capacitive</td><td>Phase 1</td></tr> <tr> <td>5</td><td>Inductive/capacitive</td><td>Phase 2</td></tr> <tr> <td>6</td><td>Inductive/capacitive</td><td>Phase 3</td></tr> <tr> <td>7</td><td>Inductive/capacitive</td><td>Total</td></tr> <tr> <td>8...15</td><td></td><td>-</td></tr> </table>	Bit	0/1	=	0	Consumption/supply	Phase 1	1	Consumption/supply	Phase 2	2	Consumption/supply	Phase 3	3	Consumption/supply	Total	4	Inductive/capacitive	Phase 1	5	Inductive/capacitive	Phase 2	6	Inductive/capacitive	Phase 3	7	Inductive/capacitive	Total	8...15		-
Bit	0/1	=																														
0	Consumption/supply	Phase 1																														
1	Consumption/supply	Phase 2																														
2	Consumption/supply	Phase 3																														
3	Consumption/supply	Total																														
4	Inductive/capacitive	Phase 1																														
5	Inductive/capacitive	Phase 2																														
6	Inductive/capacitive	Phase 3																														
7	Inductive/capacitive	Total																														
8...15		-																														
S0Pulses	WORD	<table> <tr> <td>Bit</td><td>0/1</td><td>=</td></tr> <tr> <td>0</td><td>off/on</td><td>1 per 100 digit</td></tr> <tr> <td>4</td><td>off/on</td><td>1 per 1000 digit</td></tr> <tr> <td>8</td><td>off/on</td><td>1 per 10000 digit</td></tr> </table> Referring to the energy values in the IL_4EE_MON_TERMINAL_ENERGY_VALUES structure	Bit	0/1	=	0	off/on	1 per 100 digit	4	off/on	1 per 1000 digit	8	off/on	1 per 10000 digit																		
Bit	0/1	=																														
0	off/on	1 per 100 digit																														
4	off/on	1 per 1000 digit																														
8	off/on	1 per 10000 digit																														

Tab. 9-100: Design of the IL_4EE_MON_TERMINAL_EXTENDED_VALUES structure

IL_4EE_MON_TERMINAL_POWER_VALUES

Brief description The structure contains the values of reactive and apparent powers as well as power factors measured by the power measurement terminal.

Name	Type	Description
FundamentalReactivePowerOfPhase1	REAL	Reactive power of the fundamental wave Phase 1 [VAr]
FundamentalReactivePowerOfPhase2	REAL	Reactive power of the fundamental wave Phase 2 [VAr]
FundamentalReactivePowerOfPhase3	REAL	Reactive power of the fundamental wave Phase 3 [VAr]
ApparentPowerOfPhase1	REAL	Apparent power Phase 1 [VA]
ApparentPowerOfPhase2	REAL	Apparent power Phase 2 [VA]
ApparentPowerOfPhase3	REAL	Apparent power Phase 3 [VA]

RIL_4EEEnergyManagement.library

Name	Type	Description
TotalApparentPower	REAL	Apparent power in total [VA]
PowerFactorOfPhase1	REAL	Power factor Phase 1
PowerFactorOfPhase2	REAL	Power factor Phase 2
PowerFactorOfPhase3	REAL	Power factor Phase 3
TotalPowerFactor	REAL	Power factor in total

Tab. 9-101: Design of the IL_4EE_MON_TERMINAL_POWER_VALUES structure

IL_4EE_MON_TERMINAL_ENERGY_VALUES

Brief description The structure contains the values of the active and reactive energies, separated by consumption/output and inductive/capacitive, measured by the power measurement terminal.

Name	Type	Description
RealEnergyConsumption	REAL	Active energy consumption [Wh]
RealEnergySupply	REAL	Active energy supply [Wh]
ReactiveEnergyInductive	REAL	Reactive energy inductive [VArh]
ReactiveEnergyCapacitive	REAL	Reactive energy capacitive [VArh]
OperatingHours	UDINT	Operating hours [s]

Tab. 9-102: Design of the IL_4EE_MON_TERMINAL_ENERGY_VALUES structure

IL_4EE_MON_TERMINAL_REACTIVE_POWER_VALUES

Brief description The structure contains the values of the distorted reactive powers, reactive power according to the standard and cos phi of the individual phases, measured by the power measurement terminal.

Name	Type	Description
DistortionReactivePowerOfPhase1	REAL	Distorted reactive power 1 [VAr]
DistortionReactivePowerOfPhase2	REAL	Distorted reactive power 2 [VAr]
DistortionReactivePowerOfPhase3	REAL	Distorted reactive power 3 [VAr]
ReactivePowerOfPhase1	REAL	Reactive power 1 (according to the standard) [VAr]
ReactivePowerOfPhase2	REAL	Reactive power 2 (according to the standard) [VAr]
ReactivePowerOfPhase3	REAL	Reactive power 3 (according to the standard) [VAr]
TotalApparentPower	REAL	Reactive power in total (according to the standard) [VAr]
CosPhiOfPhase1	REAL	cos(phi) of Phase 1
CosPhiOfPhase2	REAL	cos(phi) of Phase 2
CosPhiOfPhase3	REAL	cos(phi) of Phase 3

Tab. 9-103: Design of the IL_4EE_MON_TERMINAL_REACTIVE_POWER_VALUES structure

IL_4EE_MON_FLOWRATE_UNIT(ENUM)

Name	Value	Description
FLOWRATE_L_PER_SECOND	0	Flow unit is l/s
FLOWRATE_L_PER_MINUTE	1	Flow unit is l/min
FLOWRATE_L_PER_HOUR	2	Flow unit is l/h
FLOWRATE_M3_PER_SECOND	3	Flow unit is m^3/s (SI)
FLOWRATE_M3_PER_MINUTE	4	Flow unit is m^3/min
FLOWRATE_M3_PER_HOUR	5	Flow unit is m^3/h
FLOWRATE_GAL_PER_SECOND	6	Flow unit is gal/s
FLOWRATE_GAL_PER_MINUTE	7	Flow unit is gal/min
FLOWRATE_GAL_PER_HOUR	8	Flow unit is gal/h

Tab. 9-104: IL_4EE_MON_FLOWRATE_UNIT

IL_4EE_MON_PRESSURE_UNIT

Element	Value	Description
PRESSURE_UNIT_BAR	0	Unit in bar (1 Bar = 100000 Pa)
PRESSURE_UNIT_PA	1	Unit in pascal
PRESSURE_UNIT_PSI	2	Unit in psi (1 bar = 14.5 psi)

Tab. 9-105: IL_4EE_MON_PRESSURE_UNIT

IL_4EE_FLUID_MATERIAL_PARAMETER

Name	Type	Description
FluidMaterialType	IL_4EE_MON_FLUID_MATERIAL_TYPE	Type of medium, e.g. water, oil, air
FluidDensity	REAL	Medium density
FluidHeatCapacity	REAL	Heat capacity of the medium

Tab. 9-106: IL_4EE_FLUID_MATERIAL_PARAMETER

IL_4EE_MON_FLUID_MATERIAL_TYPE

Element	Value	Description
FLUID_TYPE_WATER	1	Medium "water"
FLUID_TYPE_OIL	2	Oil
FLUID_TYPE_GASOLINE	3	Gasoline
FLUID_TYPE_DRY_AIR	4	Air (dry)
FLUID_TYPE_USER_DEFINED	5	User-defined

Tab. 9-107: IL_4EE_MON_FLUID_MATERIAL_TYPE

RIL_4EEEnergyManagement.library

IL_4EE_MON_TEMPERATURE_UNIT(ENUM)**IL_4EE_MON_TEMPERATURE_UNIT(ENUM)**

Name	Value	Description
TEMPERATURE_UNIT_CELSIUS	1	Celsius °C
TEMPERATURE_UNIT_FAHRENHEIT	2	Fahrenheit °F
TEMPERATURE_UNIT_KELVIN	3	Kelvin °K

Tab. 9-108: IL_4EE_MON_TEMPERATURE_UNIT

IL_4EE_MON_GAS_TYPE(ENUM)

Name	Value	Description
GAS_TYPE_NATURAL_GAS	0	Natural gas (10kWh/m³)
GAS_TYPE_HYDROGEN	1	Hydrogen
GAS_TYPE_METHANE	2	Methane
GAS_TYPE_ETHANE	3	Ethane
GAS_TYPE_PROpane	4	Propane
GAS_TYPE_BUTANE	5	Butane
GAS_TYPE_USER_DEFINED	6	Customized, value of the "CalorificValue" input is applied

Tab. 9-109: IL_4EE_MON_GAS_TYPE

IL_4EE_CYLINDER_PARAMETER

Name	Type	Description
CylinderPistonDiameter	REAL	Piston diameter in mm
CylinderPistonRodDiameterExtension	REAL	Diameter of the piston rod in direction of extension in mm
CylinderPistonRodDiameterReturn	REAL	Diameter of the piston rod in direction of retraction in mm
StrokeLength	REAL	Stroke length in mm
ExtensionHoseDiameter	REAL	Connection diameter at extension site in mm
ReturnHoseDiameter	REAL	Connection diameter at retraction site in mm

ExtensionHoseLength	REAL	Connection length at extension site in mm
ReturnHoseLength	REAL	Connection length at retraction site in mm

Tab. 9-110: IL_4EE_CYLINDER_PARAMETER

9.2.4 HMI control data types

IL_4EE_GUI_MON_CONFIG

Brief description The IL_4EE_GUI_MON_CONFIG data structure contains the number of consumers and their configuration. The individual consumers are configured in the IL_4EE_GUI_MON_DEVICE_CONFIG substructure of the Device Array. The structure serves as the interface between the IL_4EEMonHMIDataCollectorType01 function block and the associated HMI controls. For this purpose, it has to be instantiated as st4EEMonConfig with the namespace {attribute 'namespace':='IL_4EE_Energy_Power_Monitoring'} and added to the instance of the IL_4EEMonHMIDataCollector. The structure is fully described by the function block.

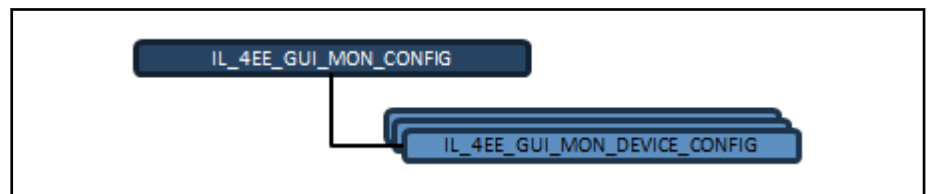


Fig. 9-45: Design of IL_4EE_GUI_MON_CONFIG structure

Name	Type	Description
NoOfDevices	DINT	Number of consumers
Device	ARRAY [1..IL_4EE_NO_OF_UNITS IL_4EE_NO_OF_DRIVES] IL_4EE_GUI_MON_DEVICE_CONFIG	+ OF Configuration of each consumer (total number calculated from number of drives and units in parameter list) see Parameter list, p. 297 , IL_4EE_GUI_MON_DEVICE_CONFIG, page 201 ,

Tab. 9-111: Min./max. and default values of the IL_4EE_GUI_MON_CONFIG structure

IL_4EE_GUI_MON_DEVICE_CONFIG

Brief description Each consumer is configured for the HMI in the IL_4EE_GUI_MON_DEVICE_CONFIG structure.

Name	Type	Description
Enable	BOOL	Consumer is active
DeviceName	STRING(16)	Name of consumer
SercosAdr	UINT	Sercos address for consumer
PowerConsumption	DINT	Average output power in [W]
DeviceType	IL_4EE_DEVICE_CONFIG_DEVICE_TYPES	Consumer type IL_4EE_DEVICE_CONFIG_DEVICE_TYPES, page 202 ,

Tab. 9-112: Min./max. and default values of the IL_4EE_GUI_MON_DEVICE_CONFIG structure

RIL_4EEEnergyManagement.library

IL_4EE_DEVICE_CONFIG_DEVICE_TYPES

Brief Description The type of consumer is specified using the IL_4EE_DEVICE_CONFIG_DEVICE_TYPES enumeration. The IL_4EEGuiMonitor function block differentiates between drives (DriveNonSercosEnergy and DriveSercosEnergy) and units (UnitNonSercosEnergy and UnitSercosEnergy).

Element	Value	Description
NonDriveNonUnit	0	Undefined, not included by the HMI
DriveNonSercosEnergy	1	Drive without Sercos energy profile
DriveSercosEnergy	2	Drive with Sercos energy profile
UnitNonSercosEnergy	3	Unit without Sercos energy profile
UnitSercosEnergy	4	Unit with Sercos energy profile

Tab. 9-113: Elements of the IL_4EE_DEVICE_CONFIG_DEVICE_TYPES enumeration type

IL_4EE_GUI_MON_RESULT_MIN

Brief description The IL_4E_GUI_MON_RESULT_MIN data structure contains the results of the IL_4EEMonHMIDataCollectorType01 function block and is connected to this function block via the VAR_IN_OUT. The structure is divided into drive (EDrive) and unit (Unit) data. The structure serves as the interface between the IL_4EEMonHMIDataCollectorType01 function block and the associated HMI controls. For this purpose, it has to be instantiated as st4EEMonResult with the namespace {attribute 'namespace':='IL_4EE_Energy_Power_Monitoring'} and added to the instance of the IL_4EEMonHMIDataCollector. The structure is fully described by the function block.

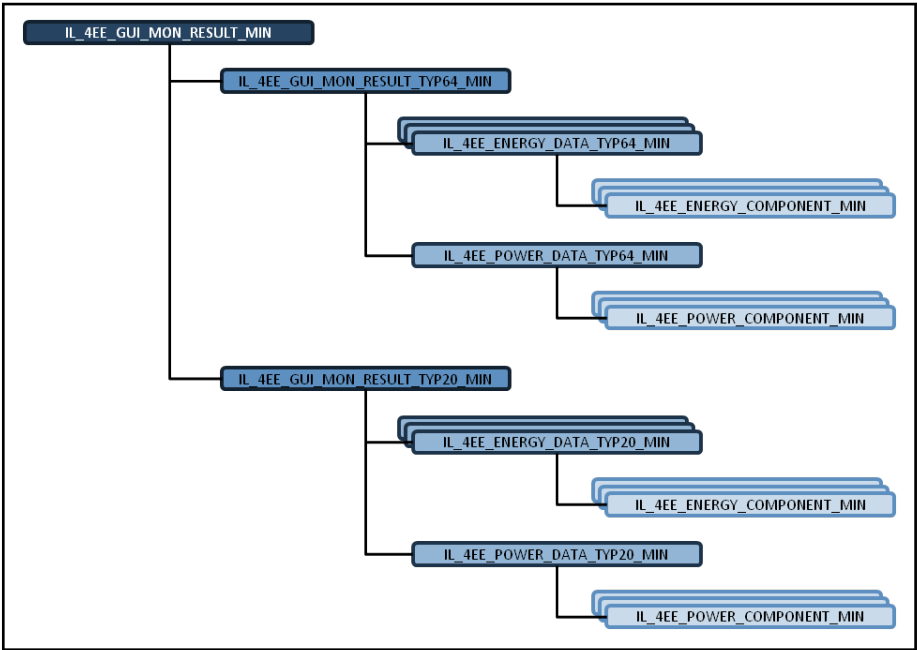


Fig. 9-46: Design of IL_4EE_GUI_MON_RESULT_MIN structure

RIL_4EEEnergyManagement.library

Name	Type	Description
EnergyMeas-Done	BOOL	Measurement complete, data current (pertains to energy data)
EnergyMeasActive	BOOL	Measurement currently running (pertains to energy data)
EDrive	IL_4EE_GUI_MON_RESULT_TYP64_MIN	The substructure contains the energy and performance data for the drives See IL_4EE_GUI_MON_RESULT_TYPE64_MIN , page 203
Unit	IL_4EE_GUI_MON_RESULT_TYP20_MIN	The substructure contains the energy and performance data for the units See IL_4EE_GUI_MON_RESULT_TYPE20_MIN , page 203

Tab. 9-114: Min./max. and default values of the IL_4EE_GUI_MON_RESULT_MIN structure

IL_4EE_GUI_MON_RESULT_TYPE64_MIN

Brief description The IL_4EE_GUI_MON_RESULT_TYPE64_MIN substructure contains the energy and performance data for the drives.

Name	Type	Description
Energy	ARRAY [1..10] OF IL_4EE_ENERGY_DATA_TYPE64_MIN	Energy data for the drives (ring memory for last 10 measurements) See IL_4EE_ENERGY_DATA_TYPE64_MIN , page 203
Power	IL_4EE_POWER_DATA_TYP64_MIN	Performance data for the drives See IL_4EE_POWER_DATA_TYPE64_MIN , page 204

Tab. 9-115: Min./max. and default values of the IL_4EE_GUI_MON_RESULT_TYPE64_MIN structure

IL_4EE_GUI_MON_RESULT_TYPE20_MIN

Brief description The IL_4EE_GUI_MON_RESULT_TYPE20_MIN substructure contains the energy and performance data for the units.

Name	Type	Description
Energy	ARRAY [1..10] OF IL_4EE_ENERGY_DATA_TYP20_MIN	Energy data for the units (ring memory for last 10 measurements) See IL_4EE_ENERGY_DATA_TYPE20_MIN , page 204
Power	IL_4EE_POWER_DATA_TYPE20_MIN	Performance data for the units See IL_4EE_POWER_DATA_TYPE20_MIN , page 204

Tab. 9-116: Min./max. and default values of the IL_4EE_GUI_MON_RESULT_TYPE20_MIN structure

IL_4EE_ENERGY_DATA_TYPE64_MIN

Brief description The IL_4EE_ENERGY_DATA_TYPE64_MIN substructure contains the energy data for the drives.

RIL_4EEEnergyManagement.library

Name	Type	Description
Component	ARRAY [1..IL_4EE_NO_OF_DRIVES] OF IL_4EE_ENERGY_COMPONENT_MIN	Energy data for the drives (ring memory for last 10 measurements) See Parameter list, page 297 See IL_4EE_ENERGY_COMPONENT_MIN, page 205

Tab. 9-117: Min./max. and default values of the IL_4EE_ENERGY_DATA_TYPE64_MIN structure

IL_4EE_POWER_DATA_TYPE64_MIN

Brief description The IL_4EE_POWER_DATA_TYPE64_MIN substructure contains the performance data for the drives.

Name	Type	Description
Component	ARRAY [1..IL_4EE_NO_OF_DRIVES] OF IL_4EE_POWER_COMPONENT_MIN	Performance data for the drives See Parameter list, page 297 See IL_4EE_POWER_COMPONENT_MIN, page 205

Tab. 9-118: Min./max. and default values of the IL_4EE_POWER_DATA_TYPE64_MIN structure

IL_4EE_ENERGY_DATA_TYPE20_MIN

Brief description The IL_4EE_ENERGY_DATA_TYPE20_MIN substructure contains the energy data for the units.

Name	Type	Description
TimeStamp	IL_4EE_TIMESTAMP_MIN	Time stamp for the measurement See IL_4EE_TIMESTAMP_MIN, page 204
Component	ARRAY [1..IL_4EE_NO_OF_UNITS] OF IL_4EE_ENERGY_COMPONENT_MIN	Energy data for the units (ring memory for last 10 measurements) See Parameter list, page 297 See IL_4EE_ENERGY_COMPONENT_MIN, page 205

Tab. 9-119: Min./max. and default values of the IL_4EE_ENERGY_DATA_TYPE20_MIN structure

IL_4EE_POWER_DATA_TYPE20_MIN

Brief description The IL_4EE_POWER_DATA_TYPE20_MIN substructure contains the performance data for the units.

Name	Type	Description
Component	ARRAY [1..IL_4EE_NO_OF_UNITS] OF IL_4EE_POWER_COMPONENT_MIN	Performance data for the units See Parameter list, page 297 See IL_4EE_POWER_COMPONENT_MIN, page 205

Tab. 9-120: Min./max. and default values of the IL_4EE_POWER_DATA_TYPE20_MIN structure

IL_4EE_TIMESTAMP_MIN

Brief description The IL_4EE_TIMESTAMP_MIN data type serves to show the start time and duration of the energy measurement.

RIL_4EEEnergyManagement.library

Name	Type	Description
Start	EXT_SYS_TIME_DATE	Start time of energy measurement
Duration	IL_4EE_DURATION_TIME	Duration of energy measurement See IL_4EE_DURATION_TIME , page 205

Tab. 9-121: Min./max. and default values of the IL_4EE_TIMESTAMP_MIN structure

IL_4EE_ENERGY_COMPONENT_MIN

Brief Description The IL_4EE_ENERGY_COMPONENT_MIN substructure contains the energy consumption of a consumer.

Name	Type	Description
Consumption	REAL	Energy consumption of a consumer in [Wh]

Tab. 9-122: Min./max. and default values of the IL_4EE_ENERGY_COMPONENT_MIN structure

IL_4EE_POWER_COMPONENT_MIN

Brief description The IL_4EE_POWER_COMPONENT_MIN substructure contains the current output power for the consumer and the time the consumer was activated. The time the output power does not equal zero is the used to determine energize time.

Name	Type	Description
Current	REAL	Current output power of consumer in [W]
Component	IL_4EE_DURATION_TIME	Time the consumer was activated See IL_4EE_DURATION_TIME , page 205

Tab. 9-123: Min./max. and default values of the IL_4EE_POWER_COMPONENT_MIN structure

IL_4EE_DURATION_TIME

Brief Description The IL_4EE_DURATION_TIME substructure serves to show the energize time for the consumer. The time is provided both in μ s and broken down into days, hours, minutes, seconds and milliseconds.

Name	Type	Description
LowValue	DWORD	Duration in μ s
HighValue	DWORD	Duration in μ s
Days	UINT	Duration in days
Hours	UINT	Duration in hours
Minutes	UINT	Duration in minutes
Seconds	UINT	Duration in seconds
Milliseconds	UINT	Duration in milliseconds

Tab. 9-124: Min./max. and default values of the IL_4EE_DURATION_TIME structure

RIL_4EEEnergyManagement.library

IL_4EE_MON_HMI_TYPE02

Brief description The IL_4EE_MON_HMI_TYPE02 structure serves as the interface between the IL_4EEMonHMIDataCollectorType02 function block and the associated web visualization.

Name	Type	Description
Constants	IL_4EE_MON_HMI_CONSTANTS_TYPE02	The constant values required to read out arrays See IL_4EE_MON_HMI_CONSTANTS_TYPE02 , page 206
Devices	IL_4EE_MON_HMI_DEVICES_TYPE02	Contains the configuration and the energy counter of the consumers as well as their time stamp See IL_4EE_MON_HMI_DEVICES_DATA_TYPE02 , page 206
Measurements	IL_4EE_MON_HMI_MEASUREMENT_TYPE02	Contains the energy measurement values See IL_4EE_MON_HMI_MEASUREMENT_TYPE02 , page 207
GroupingNames	IL_4EE_MON_HMI_STRINGS_TYPE02	Contains the strings for groupings and their groups on the HMI See IL_4EE_MON_HMI_STRINGS_TYPE02 , page 207

Tab. 9-125: Design of the IL_4EE_MON_HMI_TYPE02 structure

IL_4EE_MON_HMI_CONSTANTS_TYPE02

Brief description The structure contains the required constants to read out the different array by the HMI.

Name	Type	Description
Mode	IL_4EE_MON_HMI_MODE	Indicates to the HMI if the string for the names on the HMI are to be read out from the PLC or a file (facilitates language switching) See IL_4EE_MON_HMI_MODE , page 210
NoOfDevices	USINT	Number of consumers

Tab. 9-126: Design of the IL_4EE_MON_HMI_CONSTANTS_TYPE02 structure

IL_4EE_MON_HMI_DEVICES_TYPE02

Brief description The structure contains the time stamp of the energy counters and an array with the configuration and the energy counters of the consumers.

Name	Type	Description
Constants	TIME_OF_DAY	Time stamp of the energy counter update
Devices	ARRAY [1..NO_OF_MON_DEVICES_PER_ENERGY_MONITOR + 1] OF IL_4EE_MON_HMI_DEVICE_DATA_TYPE02	Configuration and energy counter of the consumers See IL_4EE_MON_HMI_DEVICES_DATA_TYPE02 , page 206

Tab. 9-127: Design of the IL_4EE_MON_HMI_STRINGS_TYPE02 structure

IL_4EE_MON_HMI_DEVICES_DATA_TYPE02

Brief description The structure contains the energy counter of the individual consumers.

RIL_4EEEnergyManagement.library

Name	Type	Description
Name	STRING(16)	Name of consumer
Groupings	DWORD	Group affiliation of the consumer
EnergyOdometer	REAL	Energy counter value of the consumer

Tab. 9-128: Design of the IL_4EE_MON_HMI_DEVICES_DATA_TYPE02 structure

IL_4EE_MON_HMI_MEASUREMENT_TYPE02

Brief description The structure contains the interface to operate the measurement as well as the measurement results.

Name	Type	Description
Start	BOOL	Starts the measurement
Stop	BOOL	Measurement stop
Abort	BOOL	Cancels measurement
Status	IL_4EE_MON_HMI_MEASUREMENT_STATUS	Current measurement status See IL_4EE_MON_HMI_MEASUREMENT_STATUS , page 210
BufferStart	USINT	Index of the latest energy measurement
NoOfEntries	USINT	Number of energy measurements
MeasurementResults	ARRAY[1..10] OF IL_4EE_MON_HMI_MEASUREMENT_RESULT_TYPE02	Array with energy measurements and their time stamp See IL_4EE_MON_HMI_MEASUREMENT_RESULT_TYPE02 , page 207

Tab. 9-129: Design of the IL_4EE_MON_HMI_MEASUREMENT_TYPE02 structure

IL_4EE_MON_HMI_MEASUREMENT_RESULT_TYPE02

Brief description The structure contains the measurement results and their time stamp.

Name	Type	Description
StartingTime	DATE_AND_TIME	Starting time of the measurement
Duration	TIME	Measurement duration
EnergyConsumption	ARRAY[1..NO_OF_MON_DEVICES_PER_ENERGY_MONITOR + 1] OF REAL	Measured value of the individual consumers

Tab. 9-130: Design of the IL_4EE_MON_HMI_MEASUREMENT_RESULT_TYPE02 structure

IL_4EE_MON_HMI_STRINGS_TYPE02

Brief description The structure is used to configure texts on the HMI.

Name	Type	Description
UnitEnergy	STRING(20)	Name of the energy unit for the HMI
UnitPower	STRING(20)	Name and unit of the performance for the HMI
Energy	STRING(20)	Name of the energy for the HMI
Power	STRING(20)	Name of the performance for the HMI

RIL_4EEEnergyManagement.library

Name	Type	Description
DeviceType	STRING(20)	Name of the DeviceType grouping for the HMI
EnergyType	STRING(20)	Name of the EnergyType grouping for the HMI
Grouping1	STRING(20)	Name of the grouping Grouping1
Grouping2	STRING(20)	Name of the grouping Grouping2
Grouping3	STRING(20)	Name of the grouping Grouping3
Grouping4	STRING(20)	Name of the grouping Grouping4
DeviceType-Names	ARRAY [1..64] OF STRING(20)	Names of the consumer types for the HMI (index = value of the input DeviceType +1)
EnergyType-Names	ARRAY [1..32] OF STRING(20)	Names of the energy types for the HMI (index = value of the input EnergyType +1)
Grouping1Names	ARRAY [1..64] OF STRING(20)	Names of the user-defined groups of Grouping 1 for the HMI (index = group number +1)
Grouping2Names	ARRAY [1..32] OF STRING(20)	Names of the user-defined groups of Grouping 2 for the HMI (index = group number +1)
Grouping3Names	ARRAY [1..32] OF STRING(20)	Names of the user-defined groups of Grouping 3 for the HMI (index = group number +1)
Grouping4Names	ARRAY [1..32] OF STRING(20)	Names of the user-defined groups of Grouping 4 for the HMI (index = group number +1)

Tab. 9-131: Design of the IL_4EE_MON_HMI_STRINGS_TYPE02 structure

IL_4EE_MON_HMI_TYPE03

Brief description The IL_4EE_MON_HMI_TYPE03 structure serves as the interface between the IL_4EEMonHMIDataCollectorType03 function block and the associated web visualization.

Name	Type	Description
Constants	IL_4EE_MON_HMI_CONSTANTS_TYPE03	The constant values required to read out arrays See IL_4EE_MON_HMI_CONSTANTS_TYPE03, page 208
Devices	IL_4EE_MON_HMI_DEVICES_TYPE03	Contains the configuration and the energy counter of the consumers as well as their time stamp See IL_4EE_MON_HMI_DEVICES_TYPE03, page 209
Measurements	IL_4EE_MON_HMI_MEASUREMENT_TYPE03	Contains the energy measurement values See IL_4EE_MON_HMI_MEASUREMENT_TYPE03, page 209

Tab. 9-132: Design of the IL_4EE_MON_HMI_TYPE03 structure

IL_4EE_MON_HMI_CONSTANTS_TYPE03

Brief description The structure contains the required constants to read out the different array by the HMI.

RIL_4EEEnergyManagement.library

Name	Type	Description
Mode	IL_4EE_MON_HMI_MODE	Indicates to the HMI if the string for the for the names on the HMI are to be read out from the PLC or a file (facilitates language switching) See IL_4EE_MON_HMI_MODE , page 210
NoOfDevices	USINT	Number of consumers

Tab. 9-133: Design of the IL_4EE_MON_HMI_CONSTANTS_TYPE03 structure

IL_4EE_MON_HMI_DEVICES_TYPE03

Brief description The structure contains the time stamp of the energy counters and an array with the configuration and the energy counters of the consumers.

Name	Type	Description
Constants	TIME_OF_DAY	Time stamp of the energy counter update
Devices	ARRAY [1..NO_OF_MON_DEVICES_PER_ENERGY_MONITOR + 1] OF IL_4EE_MON_HMI_DEVICE_DATA_TYPE03	Configuration and energy counter of the consumers See IL_4EE_MON_HMI_DEVICES_DATA_TYPE03 , page 209

Tab. 9-134: Design of the IL_4EE_MON_HMI_STRINGS_TYPE03 structure

IL_4EE_MON_HMI_DEVICES_DATA_TYPE03

Brief description The structure contains the energy counter of the individual consumers.

Name	Type	Description
NameID	USINT	Name of consumer
Groupings	DWORD	Group affiliation of the consumer
EnergyOdometer	REAL	Energy counter value of the consumer

Tab. 9-135: Design of the IL_4EE_MON_HMI_DEVICES_DATA_TYPE03 structure

IL_4EE_MON_HMI_MEASUREMENT_TYPE03

Brief description The structure contains the interface to operate the measurement as well as the measurement results.

Name	Type	Description
Start	BOOL	Starts the measurement
Stop	BOOL	Measurement stop
Abort	BOOL	Cancels measurement
Status	IL_4EE_MON_HMI_MEASUREMENT_STATUS	Current measurement status See IL_4EE_MON_HMI_MEASUREMENT_STATUS , page 210
BufferStart	USINT	Index of the latest energy measurement

RIL_4EEEnergyManagement.library

Name	Type	Description
NoOfEntries	USINT	Number of energy measurements
MeasurementResults	ARRAY[1..10] OF IL_4EE_MON_HMI_MEASUREMENT_RESULT_TYPE03	Array with energy measurements and their time stamp See IL_4EE_MON_HMI_MEASUREMENT_RESULT_TYPE03 , page 210

Tab. 9-136: Design of the IL_4EE_MON_HMI_MEASUREMENT_TYPE03 structure

IL_4EE_MON_HMI_MEASUREMENT_RESULT_TYPE03

Brief description The structure contains the measurement results and their time stamp.

Name	Type	Description
StartingTime	DATE_AND_TIME	Starting time of the measurement
Duration	TIME	Measurement duration
EnergyConsumption	ARRAY[1..NO_OF_MON_DEVICES_PER_ENERGY_MONITOR + 1] OF REAL	Measured value of the individual consumers

Tab. 9-137: Design of the IL_4EE_MON_HMI_MEASUREMENT_RESULT_TYPE03 structure

IL_4EE_MON_HMI_MODE

Brief description The enumeration indicates if the names for consumers, groups, etc. are read out of the control or of a configuration file.

Element	Value	Description
STRINGS_FROM_PLC	0	The names are read out of the PLC by the HMI
STRINGS_FROM_FILE	1	The names are read out of a configuration file by the HMI

Tab. 9-138: Elements of the IL_4EE_MON_HMI_MODE enumeration type

IL_4EE_MON_HMI_MEASUREMENT_STATUS

Brief description The enumeration displays the current energy measurement state.

Element	Value	Description
MEAS_IDLE	0	No measurement started
MEAS_ACTIVE	1	Measurement currently running
MEAS_DONE	2	A measurement has been completed
MEAS_ERROR	3	An error was reported in the function block

Tab. 9-139: Elements of the enumeration type IL_4EE_MON_HMI_MEASUREMENT_STATUS

9.3 EnergyControl

9.3.1 Overview

The “RIL_4EEEnergyManagement” library contains function blocks for managing and calculating energy modes for individual consumers of a machine and their visualization.

The individual function blocks are divided into the following classes:

- EnergyControl

- Command
- Device
- HMIDataCollector

The individual function blocks are connected via the declaration and communicate via mechanisms of the object-oriented extension.

Function block	Brief description
IL_4EEEnergyControlType01, page 212	The function block manages "Command" function blocks. Their specifications apply to multiple devices. Additionally, the device function block calculates the target energy mode for the registered "Device" function blocks if idle periods are specified.
IL_4EEBasicCommandType01, page 221	The function block specifies fixed energy modes (Direct Mode) and idle periods with or without subsequent switch-on. As "Command" function block, it has to be registered in an "EnergyControl" function block or in a "Device" function block (which cannot process idle periods).
IL_4EEEventCommandType01, page 228	The function block specifies fixed energy modes (Direct Mode). As "Command" function block, it has to be registered in an "EnergyControl" function block or in a "Device" function block.
IL_4EEEventCommandType02, page 235	The function block specifies fixed energy modes (Direct Mode). As "Command" function block, it has to be registered in an "EnergyControl" function block or in a "Device" function block. In contrast to IL_4EEEventCommandType02, this function block can be registered at program runtime. This is required for compiled programs.
IL_4EEBasicDeviceType01, page 242	The function block manages specifications of an "EnergyControl" function block and of the registered "Command" function blocks. It specifies the current energy mode for the consumer.
IL_4EEAdvancedDeviceType01, page 248	The function block manages specifications of an "EnergyControl" function block and of the registered "Command" function blocks. It specifies the current energy mode for the consumer. Due to the extended configuration, a function block is directly informed on idle periods.
IL_4EEEnergyFunctionType01, page 255	The function block is used to visualize analog values of an existing energy saving function (e.g. power value of a fan in percent)
IL_4EEHMIDataCollectorType01, page 257	The function block is registered in an EnergyControl function block and, for the HMI, combines data for all function blocks subordinated to the EnergyControl function block.
IL_4EEScadaDataCollector-Type01, page 259	The function block is registered in an EnergyControl function block and, for the MES/control system, combines data for all function blocks subordinated to the EnergyControl function block. Additionally, it includes "Command" function blocks for the MES/control system specifications

Tab. 9-140: Overview on the function blocks on the energy management

The structure of the function blocks is described in [fig. 9-47 "Structure of the function blocks of the energy management" on page 212](#).

An EnergyControl function block for managing and calculating the energy modes is the central component of the energy management.

The "Device" function blocks are registered via interfaces in the "EnergyControl" function blocks. The "Device" function blocks contain the configuration of the individual consumers and manage their specific energy modes. For energy management specifications that should apply to all devices, the command function blocks are registered in the "EnergyControl" function block.

RIL_4EEEnergyManagement.library

If a specification should apply for a single device, the “Command” function block can be registered in the respective “Device” function block. An “HMIDataCollector” function block can also be registered in the EnergyControl function block. It collects the data of all subordinate “Command” and “Device” function blocks required for the HMI and stores it in a structure.

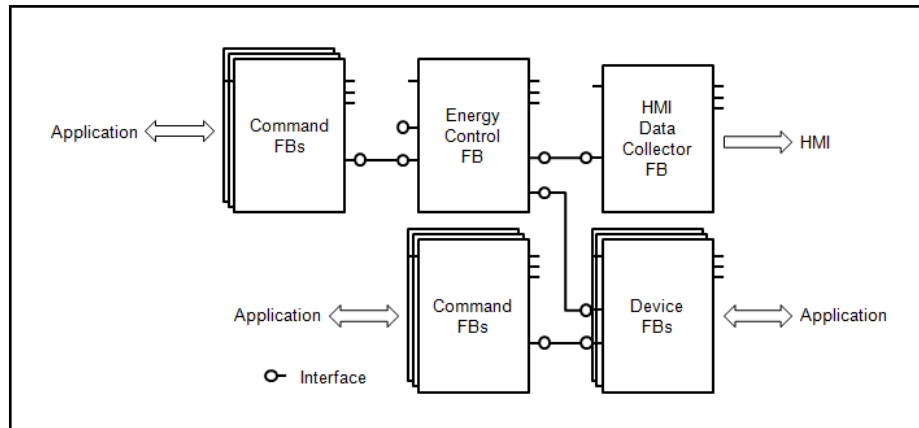
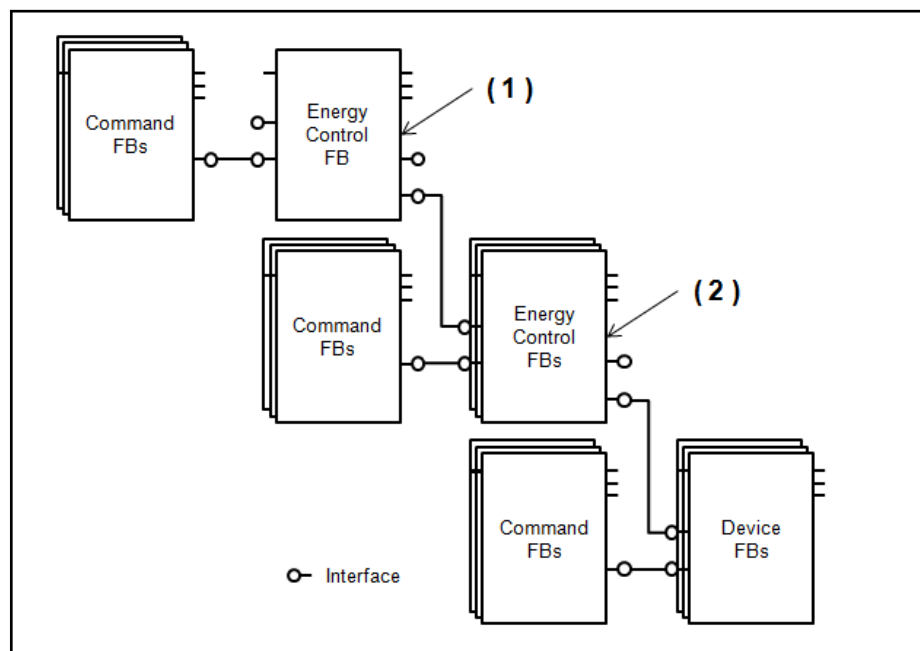


Fig. 9-47: Structure of the function blocks of the energy management

To group devices, cascade “EnergyControl” function blocks. Subordinate “EnergyControl” function blocks are registered in a parent “EnergyControl” function block. A subordinate “EnergyControl” function block is considered as device by a parent “EnergyControl” function block.



- (1) higher EnergyControl block
(2) subordinate EnergyControl blocks

Fig. 9-48: Structure of the function blocks of the energy management when cascading

9.3.2 Function Blocks and Their Methods

IL_4EEEnergyControlType01

Brief description The IL_4EEEnergyControlType01 function block is the central block of the EnergyManagement.

RIL_4EEEnergyManagement.library

It manages higher commands from command blocks that apply to multiple device blocks. To do this, both the command blocks and the device blocks are registered with the EnergyControl block via their declaration.

An HMIDataCollector can also be registered. This block determines the necessary data for showing the device and command states.

An instance of the IL_4EEEnergyControlType01 can also be registered as a device with another instance of the same block in order to form groups/modules. This allows for cascading. A higher EnergyControl block can thus be instantiated for an entire machine, with which other instances of the EnergyControl blocks are registered for modules that have to be switched individually.

The function block provides the following methods as an interface:

- mGetControl
- mSetState
- mGetState

For details, refer to [Block methods, page 214](#)

Target system	Library
MTX from 13V06	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-141: Assignment target system/library

Interface description

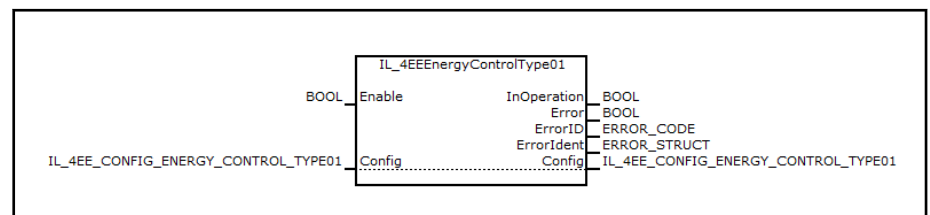


Fig. 9-49: IL_4EEEnergyControlType01 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	Config	IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01	The structure contains the necessary data for configuring the block Refer to IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01, page 263
VAR_INPUT	Enable	BOOL	Function block processing enabled (level-controlled)
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostic, see Troubleshooting, page 220

Tab. 9-142: IL_4EEEnergyControlType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 9-143: Min./max. and default values of the inputs

RIL_4EEEnergyManagement.library

Block methods The function block provides the following methods as an interface:

- [mGetControl, page 214](#)
- [mSetState, page 214](#)
- [mGetState, page 214](#)

mGetControl The mGetControl method serves to query the current command from the IL_4EEEnergyControlType01 function block.

The method should be called cyclically (e.g., 10 ms - 50 ms in consumer commanding) in order to change the energy mode specification. The higher specification from the EnergyControl block can, e.g., select the mode of operation.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	ChangeEM	IL_4EE_CHANGE_EM			Command to change the energy mode Refer to IL_4EE_CHANGE_EM, page 269
	TargetEM	USINT			EMIdent of target energy mode

Tab. 9-144: Interface for the mGetControl method with min./max. values

mSetState If the StatusOverride element is TRUE in the configuration, the current energy mode is indicated to the IL_4EEEnergyControlType01 function block via the mSetState method. The method then only has to be called after implementing a specification queried via mGetControl.

If the StatusOverride element is FALSE, this method is not needed, since the current EnergyControl block energy mode is then calculated from the feedback by the subordinate device blocks.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	CurrentEM	USINT			EMIdent of current energy mode

Tab. 9-145: Interface for the mSetState method with min./max. values

mGetState The mGetState method serves to query the current energy mode of the EnergyControl block.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	CurrentEM	USINT			EMIdent of current energy mode

Functional description The IL_4EEEnergyControlType01 block evaluates the specifications of the registered command blocks as well as any existing higher EnergyControl block. The specification with the highest priority is transmitted to the user in-

terface and device blocks. The current implementation status is returned to the device blocks.

The current specification can be queried from the application via the mGetControl method in order, e.g., to switch the mode of operation of the machine/module. The current energy mode can be queried via the mGetState method. If the current energy mode is formed in the application from the device states, it can be written via the mSetState method (see StatusOverride).

If the current energy mode is formed from the status of the device blocks, device blocks ignoring the specification from the EnergyControl block due to higher-priority command blocks are not included.

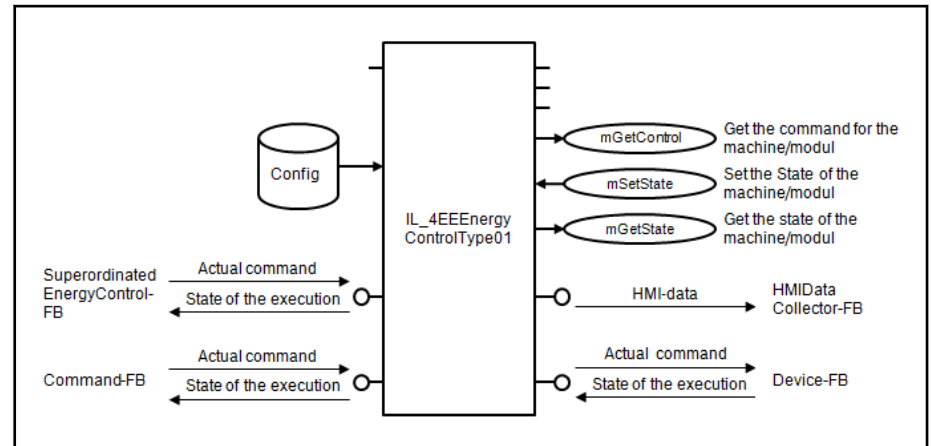


Fig. 9-50: Interface of the IL_4EEEnergyControlType01 block

Instantiating

The IL_4EEEnergyControlType01 function block belongs to the class of EnergyControl blocks. The block does not have to be registered with another block, however it can be registered with other EnergyControl blocks for cascading. command, device, and HMIdata blocks can be registered with the block.

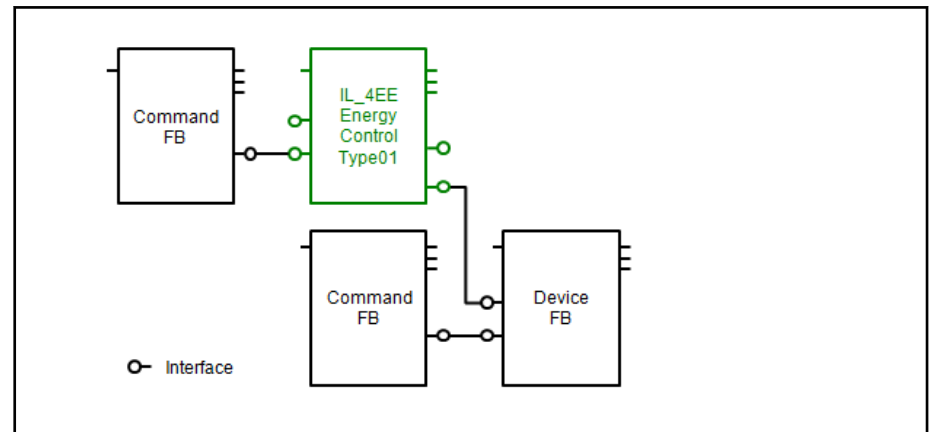


Fig. 9-51: Integrating IL_4EEEnergyControlType01 into EnergyManagement

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyControl block must be indicated as the input for the FB_Init method in the declaration of the blocks that should be registered with it. With this link, the block can communicate with the registered blocks. If the EnergyControl block should not be registered with another, the input for the FB_Init method cannot be "0".

This is the case:

- If only one EnergyControl block is used without cascading

RIL_4EEEnergyManagement.library

- With the highest EnergyControl block with cascading
SYNTAX

VAR

```
(* Instance of an EnergyControl FB, which is not registered at an other FB *)  
fbEnergyControl: IL_4EEEnergyControlType01(0);  
(* Instance of a Command FB, which is registered at the EnergyControl FB *)  
fbMachineEvent: IL_4EEEventCommandType01(fbEnergyControl);  
(* Instance of a Device FB, which is registered at the EnergyControl FB *)  
fbBasicDevice: IL_4EEBasicDeviceType01(fbEnergyControl);  
(* Instance of a Command FB, which is registered at a Device FB *)  
fbDeviceEvent: IL_4EEEventDeviceType01(fbBasicDevice);
```

END_VAR



The block registration is set by the block declaration and occurs directly after downloading the program to the control system.

It cannot be modified during the program runtime.

Maximum number of blocks registered

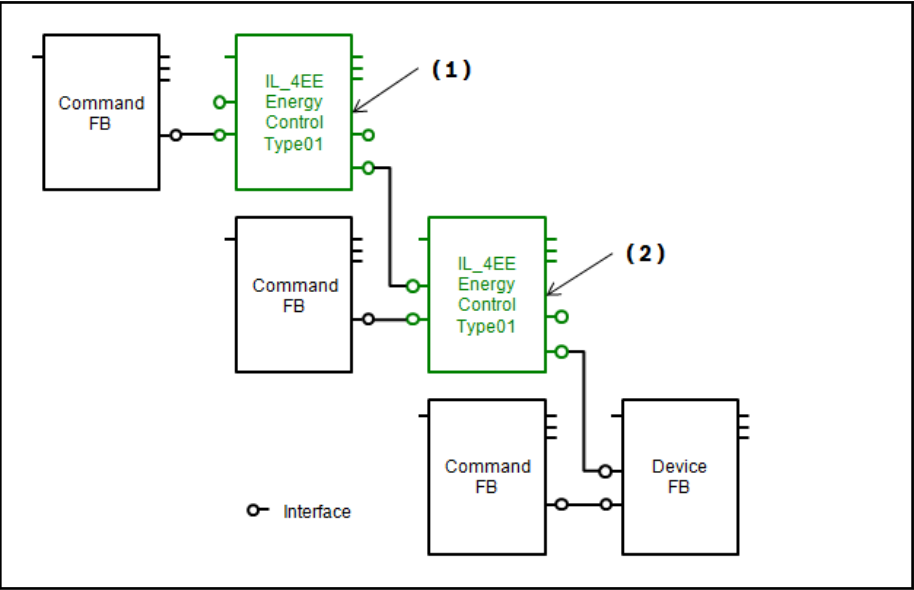
The number of command blocks that can be registered with an EnergyControl block can be modified with the constant NO_OF_COMMANDS_PER_ENERGY_CONTROL in [Parameter list, page 297](#). The default value is 16 command blocks.

The number of device blocks that can be registered with an EnergyControl block can be modified with the constant NO_OF_DEVICES_PER_ENERGY_CONTROL in [Parameter list, page 297](#). The default value is 32 device blocks.

Cascading

Type IL_4EEEnergyControlType01 blocks can be cascaded. This means that, e.g., blocks for individual machine modules can be arranged under an EnergyControl block for the entire machine. This means specifications for a machine module can also be transmitted to the EnergyManagement via command blocks.

If an EnergyControl block is registered with a higher EnergyControl block, it is treated as a device block. This means the subordinate EnergyControl block receives the specifications of the higher EnergyControl block like a device block. The subordinate EnergyControl block transmits the specifications to the device blocks registered with it.



- (1) higher EnergyControl block
- (2) subordinate EnergyControl blocks

Fig. 9-52: Cascading of IL_4EEEnergyControlType01

SYNTAX

```
VAR
(* Instance of a superordinated EnergyControl-FB, which is not
   registered at an other FB *)
fbEnergyControl01: IL_4EEEnergyControlType01(0);
(* Instance of a Command FB, which is registered at the superordinated
   EnergyControl FB *)
fbMachineEvent: IL_4EEEventCommandType01(fbEnergyControl01);
(* Instance of a subordinated EnergyControl FB, which is registered at
   the superordinate EnergyControl FB *)
fbEnergyControl02: IL_4EEEnergyControlType01(fbEnergyControl01);
(* Instance of a Command FB, which is registered at the subordinated
   EnergyControl FB *)
fbModuleEvent: IL_4EEEventCommandType01(fbEnergyControl02);
(* Instance of a Device FB, which is registered at the subordinated
   EnergyControl FB *)
fbBasicDevice: IL_4EEBasicDeviceType01(fbEnergyControl02);
(* Instance of a Command FB, which is registered at the Device FB *)
fbDeviceEvent: IL_4EEEventCommandType01(fbBasicDevice);
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

Configuration

The block is configured via the input/output structure [IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01](#), page 263..

ControlName: Specify a name for the display on the HMI here. This can be, e.g., the name of the machine or the machine module for which the block is responsible.

EMNameOP: Name of the operational energy mode for the display on the HMI.

TimeMinStayEMOp: Minimum dwell time in the operational energy mode. This can be, e.g., used to protect switching contacts. All commands are ignored during this time.

PowerConsumptionEMOp: To calculate an energy mode while specifying a dead time, consumption must be indicated while in the operational energy mode Operational. This mode is generally preset with EMIdent 16#FF.



In order to allow for a practical comparison with the other energy modes, the average consumption of the device has to be indicated when the machine is at a standstill in Operational energy mode, not the average consumption while in operation.

EMConfig: The other energy modes are configured in the substructure [IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01](#), page 264,. The following figure shows the definition of an energy mode. In addition to consumption, the time to transition between the operational energy mode and the energy mode being configured has to be indicated along with the energy needed for the transition.

RIL_4EEEnergyManagement.library

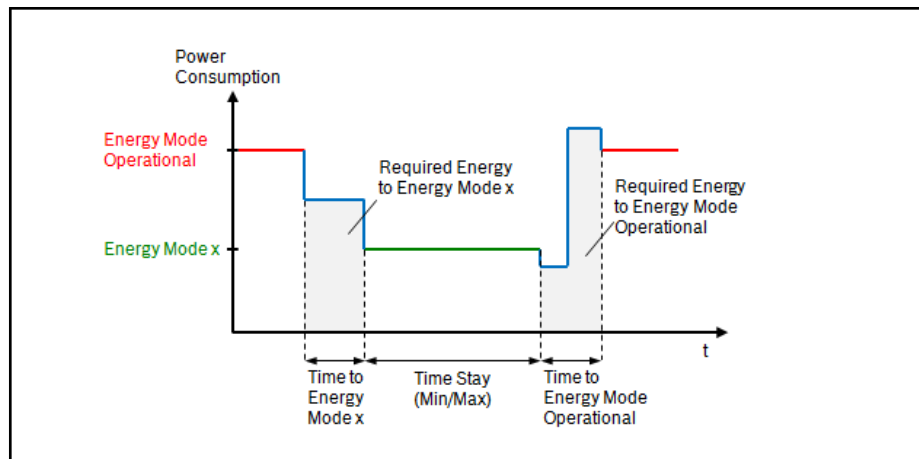


Fig. 9-53: Definition of configuration data for an IL_4EEEnergyControl/Type01 energy mode

Configure the following data for each energy mode:

- **EMValid:** The element has to have the value TRUE in order to activate the energy mode configured below
- **EMIdent:** A user-defined, coded hexadecimal value between 16#00 and 16#FE. The EMIdent 16#FF is set by default for the Operational energy mode. The energy mode is output as the target energy mode with this value by the mGetControl method
- **EMName:** Name of the energy mode for the display on the HMI
- **TimeToEMx:** Time to transition from the operational energy mode Operational to the energy mode being configured
- **TimeToEMOp:** Time to transition from the energy mode being configured to the operational energy mode Operational
- **TimeMinStay:** Minimum dwell time in the energy mode. This can be, e.g., used to protect switching contacts. All commands are ignored during this time
- **TimeMaxStay:** Maximum dwell time in the energy mode. Once this time elapses, the machine automatically switches to the operational energy mode Operational
- **PowerConsumption:** The average consumption of the device in each energy mode
- **RequEToEMx:** The required energy to transition from the operational energy mode Operational to the energy mode being configured
- **RequEToEMOp:** The required energy to transition from the energy mode being configured to the operational energy mode Operational



The number of energy modes of a device block can be modified by the constant NO_OF_EM_PER_ENERGY_CONTROL in the [Parameter list, page 297](#).

The default value is 8.

EMAssignment: The EMAssignment substructure is only needed when cascading EnergyControl blocks.

When switching to an energy mode (Direct Mode), a configured energy mode is specified from the application.

With cascading, this is also possible via a higher EnergyControl block. This is why each energy mode for the higher EnergyControl block has to be assigned an energy mode for the subordinate EnergyControl block when cascading.

Entering the EMIdent in the Config.EMAssignment array results in a table assignment:

Array Index x	Energy mode for the higher block EMAssignment[x].ControlEM	Energy mode for the subordinate block EMAssignment[x].TargetEM
0	16#FF	16#FF
1	16#A0	16#FF
2	16#80	16#A0
...		
n	16#20	16#20

Tab. 9-146: Assignment table of energy modes for subordinate and higher instance of IL_4EEEnergyControlType01 with cascading

StatusOverride: There are two options for the energy mode feedback to the EnergyControl block:

In the default instance, the statuses of the device blocks are queried and the current energy mode is formed from them.

Alternatively, the mSetState method can also be activated via the Config.StatusOverride configuration element and the current energy mode for the EnergyControl block can be formed in the application.

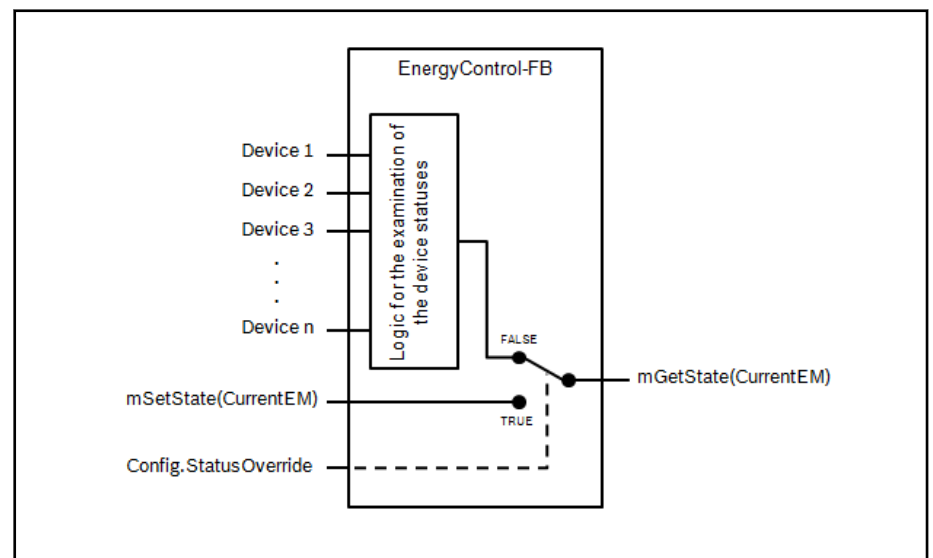


Fig. 9-54: Feedback of current energy mode to the EnergyControl block

Connecting to the application

The current EnergyControl block specification can, for example, switch the following:

- Mode of operation for the machine (without cascading/higher EnergyControl function block)
- Mode of operation for a module (subordinate EnergyControl function block for cascading)

RIL_4EEEnergyManagement.library

The mGetControl method queries the current target energy mode. In addition to the return value, it has the two outputs ChangeEM and TargetEM. The outputs are then only valid if the block is in the InOperation state. This is shown by the method return value being TRUE.

If the target energy mode changes, the ChangeEM output has the value CHANGE_ENERGY_MODE (see [IL_4EE_CHANGE_EM](#), page 269). The current energy mode is output with the TargetEM output as a coded hexadecimal value according to the configuration.

After implementing the energy mode, it can also be returned in hexadecimal code by the CurrentEM input of the mSetState method.

The ChangeEM output retains the value ChangeEnergyMode until the energy mode is returned by the device blocks is implemented (StatusOverride = FALSE) or the mSetState method (StatusOverride = TRUE), or the command block specification was aborted.

Code example:

```
...
(* Call of the method mGetControl of the instance fbEnergyControl in the
  program IL_4EE *)
bReturn:= IL_4EE.fbEnergyControl.mGetControl(ChangeEM=> enChangeEM, TargetEM=> usiTargetEM);

IF bReturn = TRUE THEN
  (* The outputs of the method are valid *)
  IF enChangeEM = ChangeEnergyMode THEN
    (* The energy mode of the output usiTargetEM has to be realised *)
    ...
    (* Return the current energy mode to the instance fbEnergyControl,
      if Config.StatusOverride = TRUE *)
    IL_4EE.fbEnergyControl.mSetState(CurrentEM:= usiCurrentEM);
  ELSE
    (* The energy mode has not to be changed *)
  END_IF
ELSE
  (* The outputs of the method are not valid *)
END_IF
...
```

Troubleshooting

If an error occurs, the specifications for all device and EnergyControl blocks registered with the instance are rescinded. The command to switch the energy mode is also rescinded (ChangeEM = NO_CHANGE). All block methods have no effect and deliver a return value of FALSE.

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR(16#0006)	16#2A10	16#0000	Cause: A configured EMIdent is greater than 16#FE Solution: Only configure EMIdents less than or equal to 16#FE
INPUT_INVALID_ERROR(16#0001)	16#2A11	16#0000	Cause: EMIdent was used in the configuration more than once Solution: Use each EMIdent only once
INPUT_INVALID_ERROR(16#0001)	16#2A11	16#0001	Cause: Energy mode assigned in EMAssignment, but not configured as EMIdent Solution: Only use configured EMIdent in EMAssignment, check EMValid

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR(16#0001)	16#2A11	16#0002	Cause: The higher EnergyControl block has an energy mode not indicated as ControlEM in Config.EMAssignment Solution: Add all EMIdents for higher EnergyControl block as ControlEM
OTHER_ERROR(16#7FFE)	16#2A12	16#0000	Cause: Device block could not be registered with a higher function block Solution: Adjust NO_OF_DEVICES_PER_ENERGY_CONTROL constant in the parameter list
STATE_MACHINE_ERROR(16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-147: Error codes of the IL_4EEEnergyControlType01 function block

IL_4EEBasicCommandType01

Brief description The IL_4EEBasicCommandType01 function block belongs to the class of command blocks. It serves to transmit commands to the EnergyManagement. This can be both the direct specification of a configured energy mode and the specification of a dead time the EnergyManagement has determined is the best energy state.

The command from the IL_4EEBasicCommandType01 block always applies to the block with which it is registered. This can be both an EnergyControl block and a device block.

The function block provides the following methods as an interface:

- mSetDirectCommand
- mSetPauseCommand
- mResetCommand
- mGetCommandState

For details, refer to [Block Methods, page 222](#)

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-148: Assignment: Target system/library

Interface description

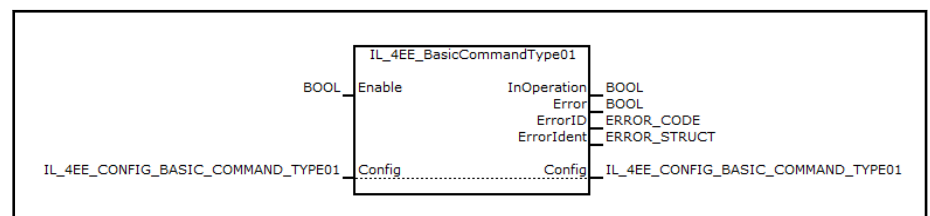


Fig. 9-55: IL_4EEBasicCommandType01 function block

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
VAR_IN_OUT	Config	IL_4EE_CONFIG_BASIC_COMMAND_TYPE01	Block configuration data See IL_4EE_CONFIG_BASIC_COMMAND_TYPE01 , page 268
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Error handling , page 228

Tab. 9-149: IL_4EEBasicCommandType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 9-150: Minimum/maximum and default values of the IL_4EEBasicCommandType01 inputs

Block Methods

The function block provides the following methods as an interface:

- [mSetDirectCommand](#), p. 222
- [mSetPauseCommand](#), p. 222
- [mResetCommand](#), p. 223
- [mGetCommandState](#), p. 223

mSetDirectCommand

The mSetDirectCommand method serves to directly specify a configured energy mode. The specification has to be rescinded by the mResetCommand method in order for a new specification to be possible.

The mGetCommandState method can query the status of the specification.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	TargetEM	USINT			Target energy mode of the command (value range: 16#00 - 16#FF)

Tab. 9-151: Interface for the mSetDirectCommand method with min./max. values

mSetPauseCommand

The mSetPauseCommand method serves to specify a dead time for which the best energy mode is calculated and implemented.

The specification has to be rescinded by the mResetCommand method in order for a new specification to be possible.

If WakeUp mode is selected, the operational energy mode is automatically switched just before dead time has elapsed so that this mode is in effect when dead time has expired.

Once the operational energy mode is set, the command is implicitly disabled.
The mGetCommandState method can query the status of the specification.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	PauseTime	TIME			Specifies dead time [s]
	WakeUpMode	BOOL			FALSE: No command is generated automatically after dead time elapses TRUE: Machine automatically switches to Operational after dead time elapses

Tab. 9-152: Interface for the mSetPauseCommand method with min./max. values

mResetCommand

The mResetCommand method serves to reset the current command from the IL_4EEBasicCommandType01 function block.

The method can be called when:

- The current command has been implemented (CommandState = CMD_INEM)
- The current command is being implemented (CommandState = CMD_ACTIVE)
- The current command is ignored due to a higher-priority command (CommandState = CMD_IGNORED)

After calling the method, the current energy mode is retained and a new command can be executed. Even commands from other command blocks with lower priority can be executed.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned

Tab. 9-153: Interface for the mResetCommand method with min./max. values

mGetCommandState

The mGetCommandState serves to query the implementation status of the command from the IL_4EEBasicCommandType01 block.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	CommandState	IL_4EE_CMD_STATE			Displays the current command state IL_4EE_CMD_STATE, page 269

Tab. 9-154: Interface for the mGetCommandState method with min./max. values

Signal-time diagram

If a valid target energy mode is specified by the mSetDirectCommand or mSetPauseCommand, the CommandState output of the mGetCommand-

RIL_4EEEnergyManagement.library

State method has the value CMD_ACTIVE. Once the energy mode is implemented, the CommandState output goes to CMD_INEM.

Calling the mResetCommand method rescinds the command and the CommandState output is reset to CMD_IDLE. Using the WakeUp mode implicitly executes the mResetCommand method once the operational energy mode is restored.

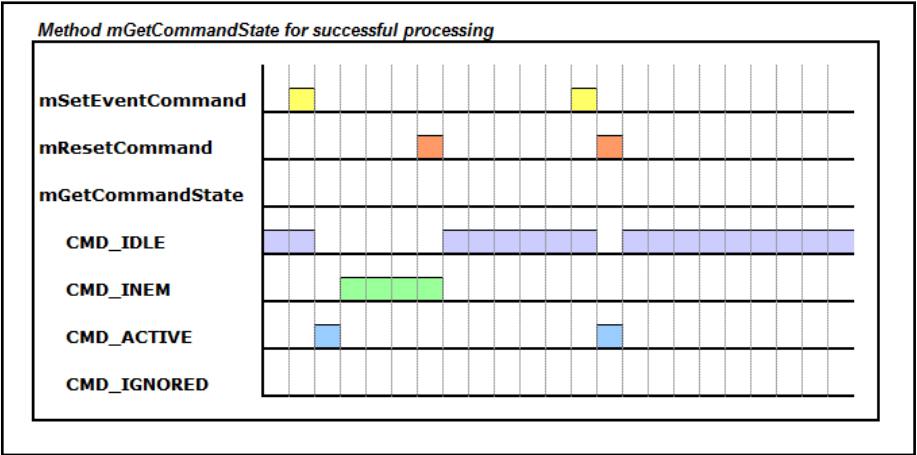


Fig. 9-56: Signal-time diagram for IL_4EEBasicCommandType01

If the command of a command block is not executed due to a higher-priority command, the CommandState output for the mGetCommandState method is CMD_IGNORED. If the higher-priority command was rescinded, the specified energy mode is implemented and the CommandState output has the value CMD_ACTIVE or CMD_INEM.

If the mResetCommand method is called while ignoring the command, the CommandState output for the mGetCommandState method is also reset to CMD_IDLE.

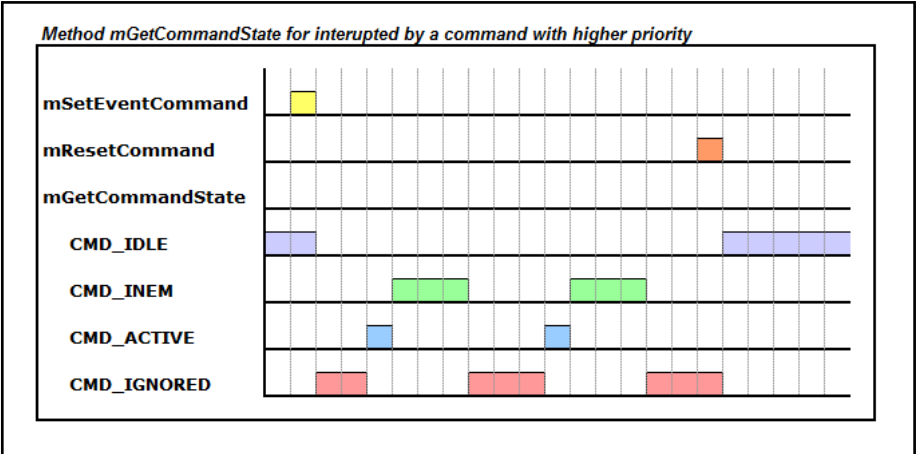


Fig. 9-57: Signal-time diagram for IL_4EEBasicCommandType01

Functional Description

The block serves to transmit commands from the application to the Energy-Management. The mSetDirectCommand and mSetPauseCommand methods are available for this.

The mSetDirectCommand method specifies a certain energy mode as an argument.

The mSetPauseCommand serves to specify dead times. This method can also be used to select automatic restart after dead time elapses. The mReset-Command method rescinds the current command.

The command is transmitted to the EnergyControl or device block with which the instance was registered. This returns the current implementation status of the command, which can be queried by the mGetCommandState method.

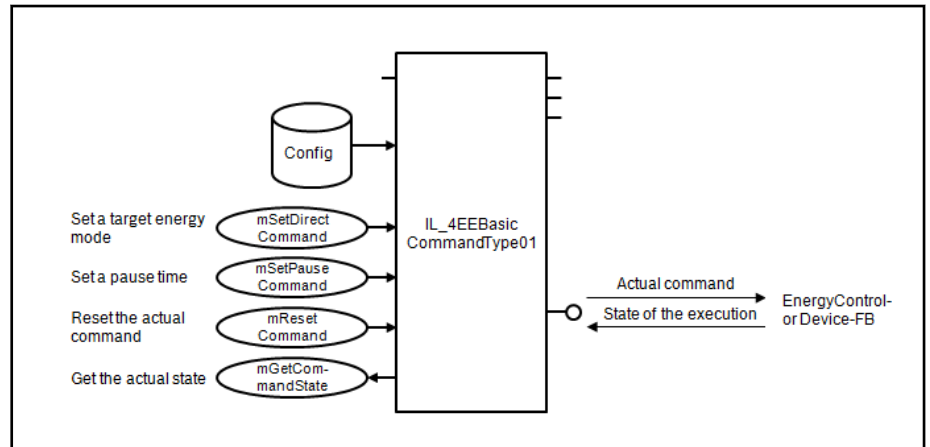


Fig. 9-58: Overview of IL_4EEBasicCommandType01 block interfaces

Instantiating

The IL_4EEBasicComandType01 function block belongs to the class of command blocks and serves to transmit commands to the EnergyManagement. Should a command only apply to a single device, the block has to be registered with the appropriate device block. Should the command apply to a group of devices, it has to be registered with each EnergyControl block.

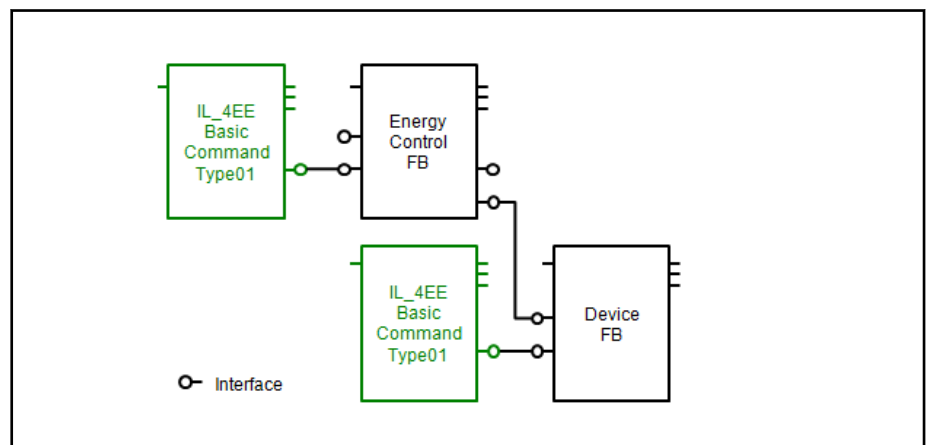


Fig. 9-59: Integrating IL_4EEBasicCommandType01 into EnergyManagement

The blocks are connected via interfaces.

In order to link the blocks together, the instance of the EnergyControl or device block with which the instance of the IL_4EEBasicCommandType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the commands to the EnergyControl or device block.

SYNTAX

VAR

```
(* Instance of the EnergyControl FB, which doesn't register itself *)
fbEnergyControl: IL_4EEEnergyControlType01(0);
(* Instance of a Command FB, which registers itself at the
   EnergyControl FB *)
fbBasicCommand1: IL_4EEBasicCommandType01(fbEnergyControl);
(* Instance of a Device FB, which registers itself at the
   EnergyControl FB *)
fbBasicDevice: IL_4EEBasicDeviceType01(fbEnergyControl);
(*Instance of a Command FB, which registers itself at the Device FB *)
```

RIL_4EEEnergyManagement.library

```
fbBasicCommand2: IL_4EEBasicCommandType01(fbBasicDevice);
END_VAR
```

Maximum Number of Blocks Registering

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

The number of command blocks that can be registered with an EnergyControl block can be modified with the constant NO_OF_COMMANDS_PER_ENERGY_CONTROL in [Parameter list, p. 297](#). The default value is 16 command blocks.

The number of command blocks that can be registered with a device block can be modified with the constant NO_OF_COMMANDS_PER_DEVICE in [Parameter list, p. 297](#). The default value is 4.

Configuration

The block is configured via the input/output structure IL_4EE_CONFIG_EXTENDED_COMMAND.

CommandName: The name of the command block is needed for the display on the HMI.

Priority: Specify the priority of the block commands as well, in order to define the valid command for a device when there are multiple commands occurring simultaneously. A value between 1 and 255 has to be entered for the priority, whereby 1 is the highest priority.

An EnergyControl block transmits the priority of its valid command to the subordinate device blocks. The device block compares them with the pending commands of directly registered command blocks and executes the highest-priority command.

Example:

If the command block registered with the EnergyControl block in [fig. 9-59 "Integrating IL_4EEBasicCommandType01 into EnergyManagement" on page 225](#) has a priority of 50 and the command block registered with the device block has a priority of 60, the command from the command block registered with the EnergyControl block is executed if there are simultaneous commands.

The command block registered with the device block returns Ignore via the mGetCommandState method.

Connecting to the Application

For EnergyControl block commands, the block has the mSetDirectCommand and mSetPauseCommand methods. The mSetDirectCommand method specifies a certain energy mode. This has to be configured for the block with which the command block is registered. The mResetCommand method can abort the transition to another energy mode. This has to be taken into consideration when the consumers are switching. The mSetState method completes the transition to the energy mode by the feedback to the device or EnergyControl block. If the specified energy mode is reached, the command can be reset by the mResetCommand method. This allows commands with lower priority to be executed.

The mSetPauseCommand method specifies a dead time for which the optimal energy mode is calculated. A WakeUp mode can also be activated optionally.

If WakeUp mode is not selected, the behavior is identical to a command from mSetDirectCommand. In order to switch to the operational energy mode once dead time has elapsed in this instance, mSetDirectCommand has to issue the appropriate command.

If WakeUp mode is selected, the switch to operational energy mode is automatically executed before dead time elapses so that the operational energy mode is set at the end of dead time. In this instance, the mResetCommand method is implicitly called and ends the command.

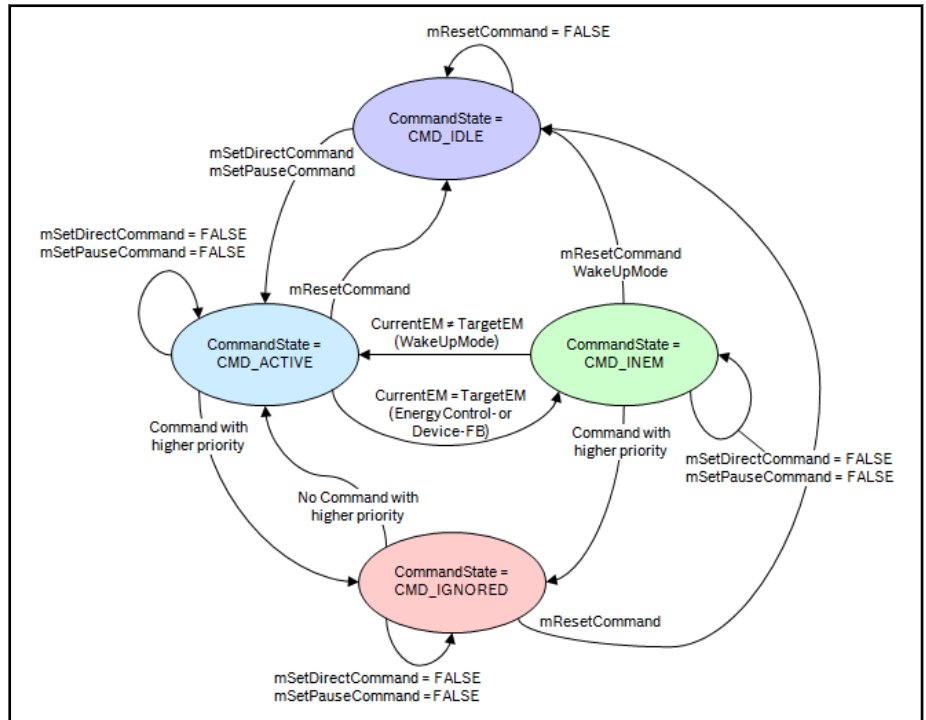


Fig. 9-60: Machine state for commands from IL_4EEBasicCommandType01 to the EnergyManagement

The mGetCommandState method can query the status of the command block command.

Sample code:

```

...
(* The variable bChangeEM has to be set by the application if the command
is to be set *)
IF bPause = TRUE THEN
    (* Call of the method mSetPauseCommand of the instance fbBasicCommand
    in the program IL_4EE, the WakeUpMode is not activated, that means
    that after the pause time is elapsed, the energy mode "Operational"
    will not be forced automatically *)
    IL_4EE.fbBasicCommand.mSetPauseCommand(PauseTime:= udiPauseTime, WakeUpMode:= FALSE);
    (* Optional reset of the variable *)
    bPause := FALSE;
END_IF
...
(* The variable bEndPause has to be set by the application if the command
is to be reset *)
IF bEndPause = TRUE THEN
    (* Call of the method mResetCommand, of the instanz fbBasicCommand in
    the programm IL_4EE, the command gets reset *)
    IL_4EE.fbBasicCommand.mResetCommand();
    (* Optional reset of the variable *)
    bEndPause:= FALSE;
END_IF
...
(* By calling the method mGetCommandState the state of the actual command
can be checked *)
IL_4EE.fbBasicCommand.mGetCommandState(InEM=> bInEM, CommandState=> enCmdState)

CASE enCmdState OF
    NO_CMD:
        (* No actual command *)
    CMD_ACTIVE:
        (* The command is in realisation *)

```

RIL_4EEEnergyManagement.library

```

CMD_INEM:
  (*The command is still actual and has been realised (target energy
  mode has been reached) *)
CMD_IGNORED:
  (* The command is still actual, but is ignored because of a command
  with higher priority *)
ELSE
  (* Invalid return value *)
END_CASE
...

```



Make sure to also reset a command in the operational energy mode with the mResetCommand method. Otherwise lower-priority commands cannot be executed by the appropriate EnergyControl or device block.

Error handling

If an error occurs in the block, the commands currently pending are reset. All block methods have no effect and deliver a return value of FALSE.

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A10	16#0001	Cause: The priority cannot be less than 1 Solution: Set priority to a value greater than 0
OTHER_ERROR (16#7FFE)	16#2A12	16#0001	Cause: Could not register as command block with a higher function block Solution: Adjust NO_OF_COMMANDS_PER_ENERGY_CONTROL or NO_OF_COMMANDS_PER_DEVICE constant in the parameter list
STATE_MACHINE_ERROR (16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-155: Error codes of the IL_4EEBasicCommandType01 function block

IL_4EEEventCommandType01**Brief description**

The IL_4EEEventCommandType01 function block belongs to the class of command blocks. It serves to transmit the direct specifications of a configured energy mode to the EnergyManagement.

What separates this block from IL_4EEBasicCommandType01 is that a delay time can be specified. The command is only transmitted to the device or EnergyControl block once this time has elapsed without the command being rescinded. This is particularly recommended for automated events from the application (e.g., lack of parts), since this can ensure a minimal duration of the corresponding state.

The command from the IL_4EEEventCommandType01 block always applies to the block with which it is registered. This can be both an EnergyControl block and a device block.

The function block provides the following methods as an interface:

- mSetEventCommand
- mResetCommand
- mGetCommandState

For details, refer to [Block Methods, page 229](#)

Target system	Library
MTX from 13V06	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-156: Assignment: Target system/library

Interface description

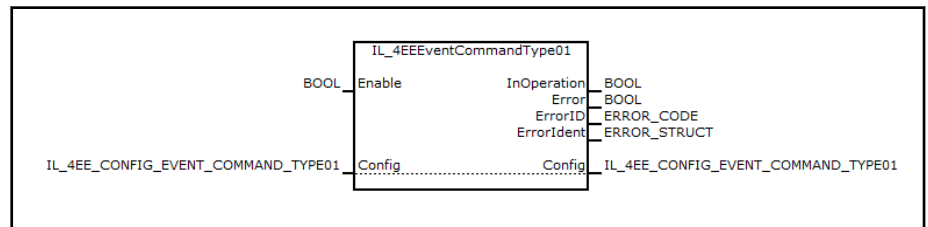


Fig. 9-61: IL_4EEEventCommandType01 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	Config	IL_4EE_CONFIG_EVENT_COMMAND_TYPE01	The structure contains the necessary data for configuring the block. See IL_4EE_CONFIG_EVENT_COMMAND_TYPE01 , page 265
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostic, see Error handling , page 235

Tab. 9-157: IL_4EEEventCommandType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 9-158: Minimum/maximum and default values of the IL_4EEEventCommandType01 inputs

Block Methods

The function block provides the following methods as an interface:

- [mSetEventCommand](#), p. 229
- [mResetCommand](#), p. 230
- [mGetCommandState](#), p. 230

mSetEventCommand

The mSetEventCommand method serves to directly specify a configured energy mode. The specification has to be rescinded by the mResetCommand method in order for a new specification to be possible.

The mGetCommandState method can query the status of the specification.

RIL_4EEEnergyManagement.library

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned

mResetCommand

Tab. 9-159: Interface for the mSetEventCommand method with min./max. values

The mResetCommand method serves to reset the current command from the IL_4EEEventCommandType01 function block.

The method can be called when:

- The current command has been implemented (CommandState = CMD_INEM)
- The current command is being implemented (CommandState = CMD_ACTIVE)
- The current command is ignored due to a higher-priority command (CommandState = CMD_IGNORED)

After calling the method, the current energy mode is retained and a new command can be executed. Even commands from other command blocks with lower priority can be executed.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned

mGetCommandState

Tab. 9-160: Interface for the mResetCommand method with min./max. values

The mGetCommandState serves to query the implementation status of the command from the IL_4EEEventCommandType01 block.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	RemainingTime	TIME			Remaining time [s] until the TargetEM is specified
	CommandState	IL_4EE_CMD_STATE			Displays the current command state See IL_4EE_CMD_STATE , page 269

Tab. 9-161: Interface for the mGetCommandState method with min./max. values

Signal-time diagram

If a target energy mode is specified by the mSetEventCommand, the CommandState output of the mGetCommandState method has the value CMD_ACTIVE. Once the energy mode is implemented, the CommandState output goes to CMD_INEM.

Calling the mResetCommand method rescinds the command and the CommandState output is reset to CMD_IDLE.

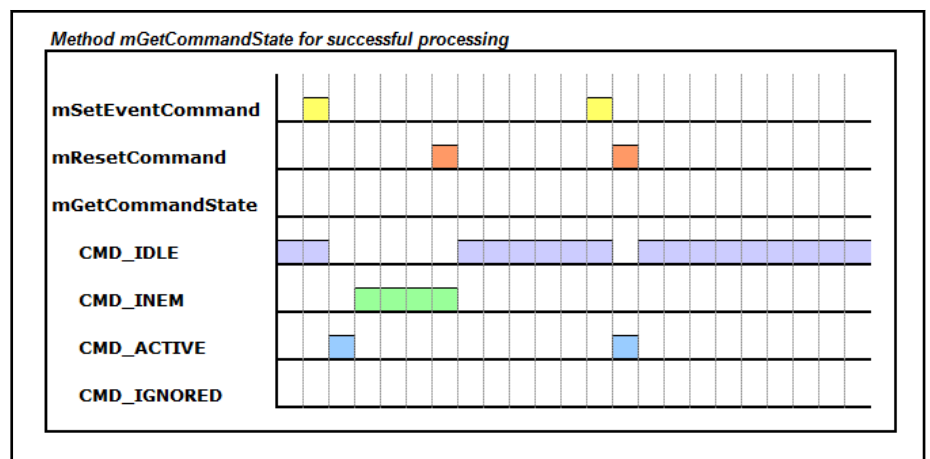


Fig. 9-62: Signal-time diagram for IL_4EEEventCommandType01

If the command of a command block is not executed due to a higher-priority command, the CommandState output for the mGetCommandState method is CMD_IGNORED. If the higher-priority command was rescinded, the specified energy mode is implemented and the CommandState output has the value CMD_ACTIVE or CMD_INEM.

If the mResetCommand method is called while ignoring the command, the CommandState output for the mGetCommandState method is also reset to CMD_IDLE.

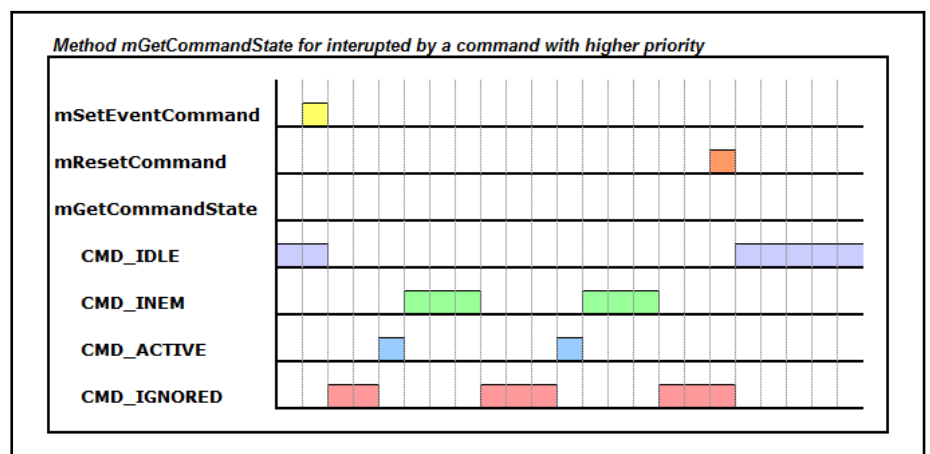


Fig. 9-63: Signal-time diagram for IL_4EEEventCommandType01

Functional description

The block serves to transmit commands from the application to the Energy-Management. The mSetEventCommand method is available for this. The mSetEventCommand method specifies the energy mode configured for the block. The mResetCommand method rescinds the current command.

The command is transmitted to the EnergyControl or device block with which the instance was registered. This returns the current implementation status of the command, which can be queried by the mGetCommandState method.

RIL_4EEEnergyManagement.library

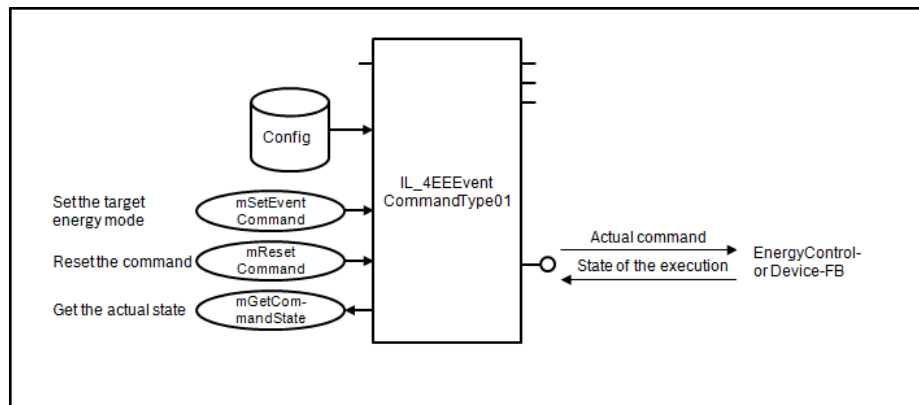


Fig. 9-64: Overview of IL_4EEEventCommandType01 block interfaces

Instantiating

The IL_4EEEventComandType01 function block belongs to the class of command blocks and serves to transmit commands to the EnergyManagement. Should a command only apply to a single device, the block has to be registered with the appropriate device block. Should the command apply to a group of devices, it has to be registered with each EnergyControl block.

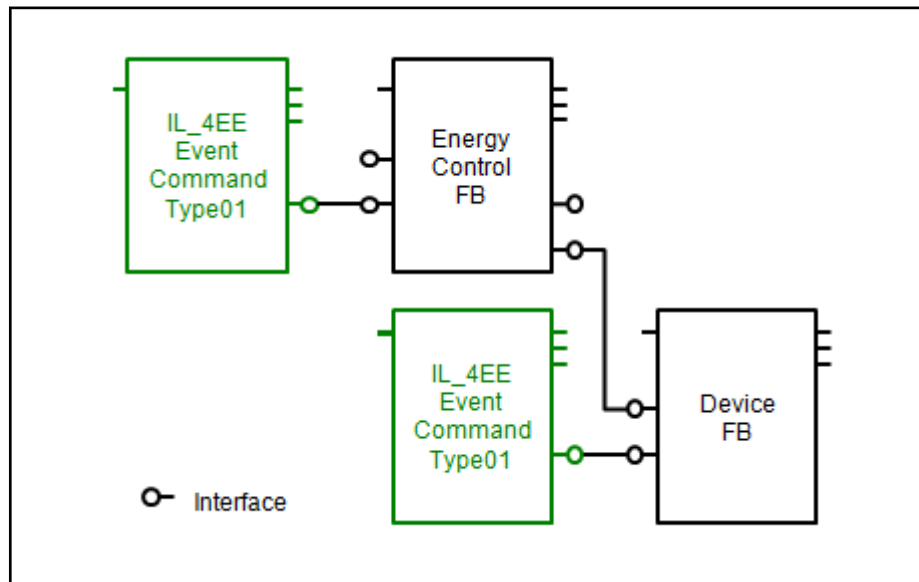


Fig. 9-65: Integrating IL_4EEEventCommandType01 into EnergyManagement

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyControl or device block with which the instance of the IL_4EEEventCommandType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the commands to the EnergyControl or device block.

SYNTAX

VAR

```
(* Instance of the EnergyControl FB, which doesn't register itself *)
fbEnergyControl: IL_4EEEnergyControlType01(0);
(* Instance of a Command FB, which registers itself at the
EnergyControl FB *)
fbMachineEvent: IL_4EEEventCommandType01(fbEnergyControl);
(* Instance of a Device FB, which registers itself at the
EnergyControl FB *)
fbBasicDevice: IL_4EEBasicDeviceType01(fbEnergyControl);
(* Instance of a Command FB, which registers itself at the Device FB *)
fbDeviceEvent: IL_4EEEventCommandType01(fbBasicDevice);
```

END_VAR

	The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.
Maximum Number of Blocks Registering	The number of command blocks that can be registered with an EnergyControl block can be modified with the constant NO_OF_COMMANDS_PER_ENERGY_CONTROL in Parameter list, p. 297,. The default value is 16. The number of command blocks that can be registered with a device block can be modified with the constant NO_OF_COMMANDS_PER_DEVICE in Parameter list, p. 297,. The default value is 4.
Configuration	The block is configured via the input/output structure IL_4EE_CONFIG_EVENT_COMMAND_TYPE01, p. 265. CommandName: The name of the command block is needed for the display on the HMI. TargetEM: The target energy mode in the command is set in the configuration for the IL_4EEEventCommandType02 block. DelayTime: The DelayTime structure element serves to configure a delay time for the command block. The command is only active once this time has elapsed. This can also rescind the command during this time without Energy-Management reacting. This is particularly recommended if a minimum duration should be specified for automatically generated commands from the application. Priority: Specify the priority of the block commands as well, in order to define the valid command for a device when there are multiple commands occurring simultaneously. A value between 1 and 255 has to be entered for the priority, whereby 1 is the highest priority. An EnergyControl block transmits the priority of its valid command to the subordinate device blocks. The device block compares them with the pending commands of directly registered command blocks and executes the highest-priority command. Example: If the command block registered with the EnergyControl block in fig. 9-65 "Integrating IL_4EEEventCommandType01 into EnergyManagement" on page 232 has a priority of 50 and the command block registered with the device block has a priority of 60, the command from the command block registered with the EnergyControl block is executed if there are simultaneous commands. The command block registered with the device block returns Ignore via the mGetCommandState method.
Connecting to the Application	The mSetEventCommand method specifies the target energy mode. This has to be configured for the block with which the command block is registered. The mResetCommand method can abort the transition to another energy mode. This has to be taken into consideration when the consumers are switching. The mSetState method completes the transition to the energy mode by the feedback to the device or EnergyControl block. If the specified energy mode is reached, the command can be reset by the mResetCommand method. This allows commands with lower priority to be executed.

RIL_4EEEnergyManagement.library

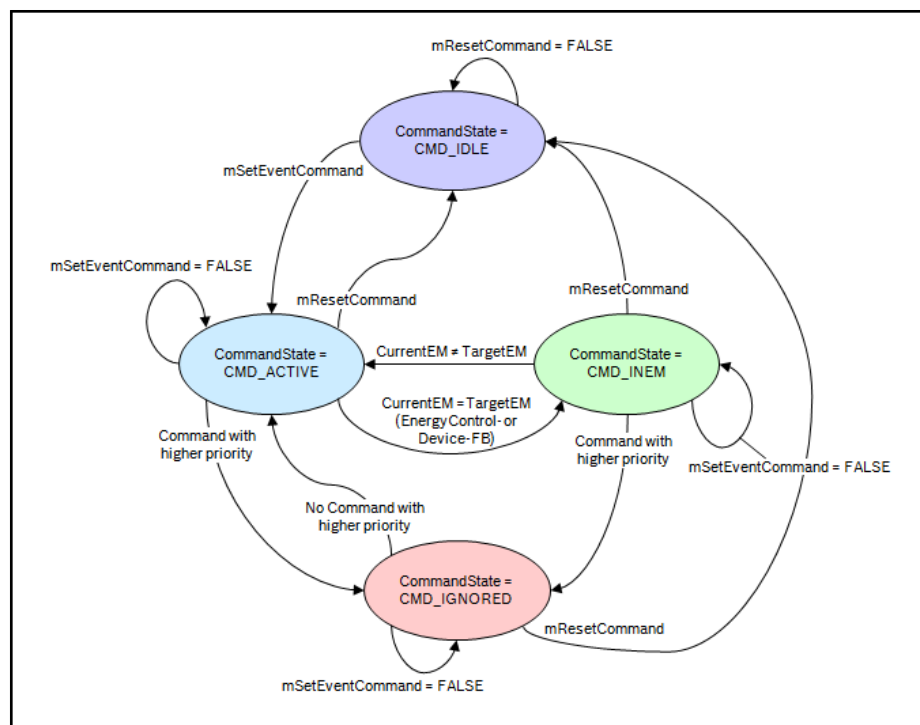


Fig. 9-66: Machine state for commands from IL_4EEEventCommandType01 to the Energy Management

The mGetCommandState method can query the status of the command block command.

Sample code:

```

...
(* The variable bChangeEM has to be set by the application if the command
is to be set *)
IF bChangeEM = TRUE THEN
  (* Call of the method mSetEventCommand of the instance fbEventCommand
  in the program IL_4EE *)
  IL_4EE.fbEventCommand.mSetDirectCommand(TargetEM:=> uiTargetEM);
  (* Optional reset the pause condition *)
  bChangeEM := FALSE;
END_IF
...
(* The variable bEndPause has to be set in the Application, if the pause
condition is obsolete *)
IF bEndPause = TRUE THEN
  (* Call of the method mResetCommand of the instance mResetCommand in
  the program IL_4EE, the command gets reset *)
  IL_4EE.fbEventCommand.mResetCommand();
  (* Optional reset the end condition *)
  bEndPause:= FALSE;
END_IF
...
(* By calling the method mGetCommandState the state of the actual command
can be checked *)
IL_4EE.fbEventCommand.mGetCommandState(RemainingTime=> uiRemainingTime, CommandState=> enCmdState)
CASE enCmdState OF
  NO_CMD:
    (* No actual command *)
  CMD_ACTIVE:
    (* The command is in realisation *)
  CMD_INEM:
    (* The command is still actual and has been realised (target energy
    mode has been reached *)
  CMD_IGNORED:
    (* The command is still actual, but is ignored because of a command
    with higher priority *)
ELSE
  (* Invalid return value *)
END_CASE
...

```


END_IF
...

Make sure to also reset a command in the operational energy mode with the mResetCommand method. Otherwise no lower-priority commands can be executed.

Error handling

If an error occurs in the block, the commands currently pending are reset. All block methods have no effect and deliver a return value of FALSE.

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A10	16#0001	Cause: The priority cannot be less than 1 Solution: Set priority to a value greater than 0
INPUT_INVALID_ERROR (16#0001)	16#2A11	16#0003	Cause: Higher block does not have TargetEM Solution: Indicate a TargetEM that exists as the EMI- dent for the higher block
OTHER_ERROR (16#7FFE)	16#2A12	16#0001	Cause: Could not register as command block with a higher function block Solution: Adjust NO_OF_COMMANDS_PER_ENER- GY_CONTROL or NO_OF_COMMANDS_PER_DE- VICE constant in the parameter list
STATE_MACHINE_ERROR (16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-162: Error codes of the IL_4EEEventCommandType01 function blocks

IL_4EEEventCommandType02

Brief description

The IL_4EEEventCommandType02 function block belongs to the class of command blocks. It serves to transmit the direct specifications of a configured energy mode to the EnergyManagement.

Unlike the function block [IL_4EEEventCommandType01, p. 228](#), this is registered with the EnergyControl or command block for runtime. This is recommended if an array of command blocks should be registered with an array of device blocks.

What separates this block from IL_4EEBasicCommandType01 is that a delay time can be specified.

The command is only transmitted to the device or EnergyControl block once this time has elapsed without the command being rescinded. This is particularly recommended for automated events from the application (e.g., lack of parts), since this can ensure a minimal duration of the corresponding state.

The command from the IL_4EEEventCommandType01 block always applies to the block with which it is registered. This can be both an EnergyControl block and a device block.

The function block provides the following methods as an interface:

- mSetEventCommand
- mResetCommand
- mGetCommandState

For details, refer to [Block Methods, page 236](#)

RIL_4EEEnergyManagement.library

Target system	Library
MTX from 13V06	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-163: Assignment: Target system/library

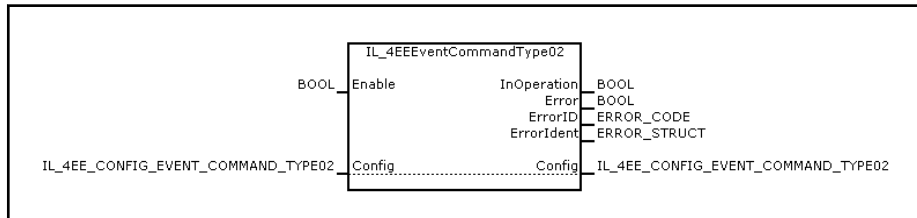


Fig. 9-67: IL_4EEEventCommandType02 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	Config	IL_4EE_CONFIG_EVENT_COMMAND_TYPE01	The structure contains the necessary data for configuring the block. See IL_4EE_CONFIG_EVENT_COMMAND_TYPE01 , page 265
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
	Command-Server	ITF_Device	EnergyControl or device block with which the instance should be registered
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostic, see Error handling , page 242

Tab. 9-164: IL_4EEEventCommandType02 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 9-165: Minimum/maximum and default values of the IL_4EEEventCommandType02 inputs

Block Methods

The function block provides the following methods as an interface:

- [mSetEventCommand](#), p. 236
- [mResetCommand](#), p. 237
- [mGetCommandState](#), p. 237

mSetEventCommand

The mSetEventCommand method serves to directly specify a configured energy mode. The specification has to be rescinded by the mResetCommand method in order for a new specification to be possible.

The mGetCommandState method can query the status of the specification.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned

Tab. 9-166: Interface for the mSetEventCommand method with min./max. values

mResetCommand

The mResetCommand method serves to reset the current command from the IL_4EEEventCommandType02 function block.

The method can be called when:

- The current command has been implemented (CommandState = CMD_INEM)
- The current command is being implemented (CommandState = CMD_ACTIVE)
- The current command is ignored due to a higher-priority command (CommandState = CMD_IGNORED)

After calling the method, the current energy mode is retained and a new command can be executed. Even commands from other command blocks with lower priority can be executed.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned

Tab. 9-167: Interface for the mResetCommand method with min./max. values

mGetCommandState

The mGetCommandState serves to query the implementation status of the command from the IL_4EEEventCommandType02 block.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	RemainingTime	TIME			Remaining time [s] until the TargetEM is specified
	CommandState	IL_4EE_CMD_STATE			Displays the current command state See IL_4EE_CMD_STATE , page 269

Tab. 9-168: Interface for the mGetCommandState method with min./max. values

Signal-time diagram

If a valid target energy mode is specified by the mSetEventCommand or mSetPauseCommand, the CommandState output of the mGetCommandState method has the value CMD_ACTIVE. Once the energy mode is implemented, the CommandState output goes to CMD_INEM.

Calling the mResetCommand method rescinds the command and the CommandState output is reset to CMD_IDLE.

RIL_4EEEnergyManagement.library

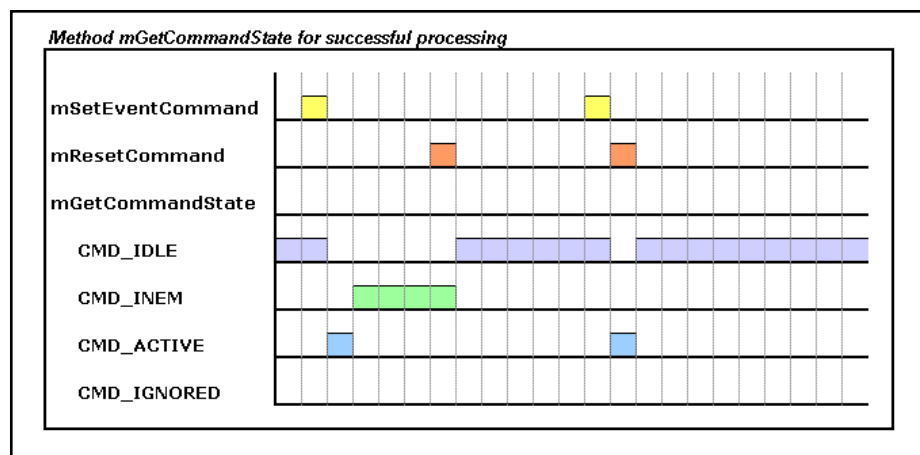


Fig. 9-68: Signal-time diagram for IL_4EEEventCommandType0

If the command of a command block is not executed due to a higher-priority command, the CommandState output for the mGetCommandState method is CMD_IGNORED. If the higher-priority command was rescinded, the specified energy mode is implemented and the CommandState output has the value CMD_ACTIVE or CMD_INEM.

If the mResetCommand method is called while ignoring the command, the CommandState output for the mGetCommandState method is also reset to CMD_IDLE.

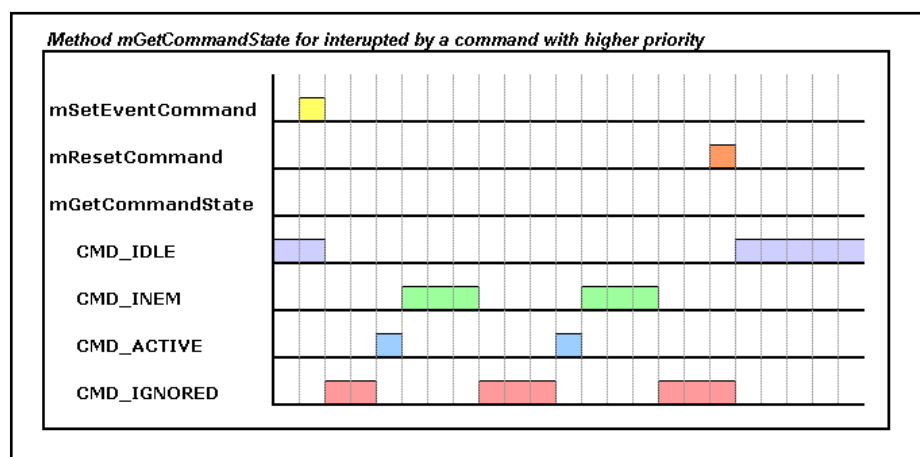


Fig. 9-69: Signal-time diagram for IL_4EEEventCommandType02

Functional Description

The block serves to transmit commands from the application to the Energy-Management. The mSetEventCommand method is available for this. The mSetEventCommand method specifies the energy mode configured for the block. The mResetCommand method rescinds the current command.

The command is transmitted to the EnergyControl or device block with which the instance was registered. This returns the current implementation status of the command, which can be queried by the mGetCommandState method.

Compared to the IL_4EEEventCommandType01 function block, the Energy-Control or device block is not registered by the FB_Init method in the declaration, but as the input for the block.

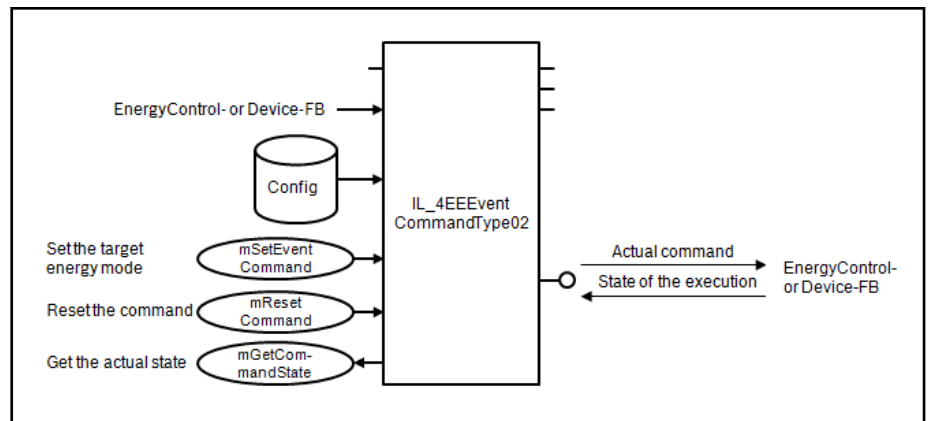


Fig. 9-70: Overview of IL_4EEEventCommandType02 block interfaces

Instantiating

The IL_4EEEventComandType02 function block belongs to the class of command blocks and serves to transmit commands to the EnergyManagement. Should a command only apply to a single device, the block has to be registered with the appropriate device block. Should the command apply to a group of devices, it has to be registered with each EnergyControl block.

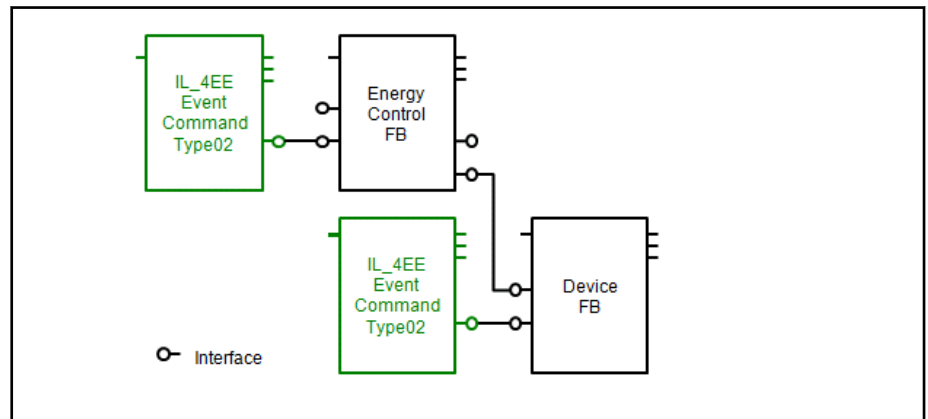


Fig. 9-71: Integrating IL_4EEEventCommandType02 into EnergyManagement

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyControl or device block with which the instance of the IL_4EEEventCommandType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can register and transmit the commands to the EnergyControl or device block.

SYNTAX

```
VAR
(* Instance of an EnergyControl FB, which doesn't register itself *)
fbEnergyControl: IL_4EEEnergyControlType01(0);
(* Instance of a Command FB *)
fbMachineEvent: IL_4EEEventCommandType02;
(* Instance of a Device FB, which registers itself at a EnergyControl FB *)
fbBasicDevice: IL_4EEBasicDeviceType01(fbEnergyControl);
(* Instance of a Command-FB *)
fbDeviceEvent: IL_4EEEventCommandType02;
END_VAR

(* The Command FB registers itself at the EnergyControl FB *)
fbMachineEvent(CommandServer := fbEnergyControl, Config := stMachineEventConfig);
(* The Command FB registers itself at the Device FB *)
fbDeviceEvent(CommandServer := fbBasicDevice, Config := stMachineEventConfig);
```

RIL_4EEEnergyManagement.library

	The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.
Maximum Number of Blocks Registering	The number of command blocks that can be registered with an EnergyControl block can be modified with the constant NO_OF_COMMANDS_PER_ENERGY_CONTROL in Parameter list, p. 297,. The default value is 16. The number of command blocks that can be registered with a device block can be modified with the constant NO_OF_COMMANDS_PER_DEVICE in Parameter list, p. 297,. The default value is 4.
Configuration	The block is configured via the input/output structure IL_4EE_CONFIG_EVENT_COMMAND_TYPE01, p. 265. CommandName: The name of the command block is needed for the display on the HMI. TargetEM: The target energy mode in the command is set in the configuration for the IL_4EEEventCommandType02 block. DelayTime: The DelayTime structure element serves to configure a delay time for the command block. The command is only active once this time has elapsed. This can also rescind the command during this time without Energy-Management reacting. This is particularly recommended if a minimum duration should be specified for automatically generated commands from the application. Priority: Specify the priority of the block commands as well, in order to define the valid command for a device when there are multiple commands occurring simultaneously. A value between 1 and 255 has to be entered for the priority, whereby 1 is the highest priority. An EnergyControl block transmits the priority of its valid command to the subordinate device blocks. The device block compares them with the pending commands of directly registered command blocks and executes the highest-priority command. Example: If the command block registered with the EnergyControl block in fig. 9-71 "Integrating IL_4EEEventCommandType02 into EnergyManagement" on page 239 has a priority of 50 and the command block registered with the device block has a priority of 60, the command from the command block registered with the EnergyControl block is executed if there are simultaneous commands. The command block registered with the device block returns Ignore via the mGetCommandState method.
Connecting to the Application	The mSetEventCommand method specifies the target energy mode. This has to be configured for the block with which the command block is registered. The mResetCommand method can abort the transition to another energy mode. This has to be taken into consideration when the consumers are switching. The mSetState method completes the transition to the energy mode by the feedback to the device or EnergyControl block. If the specified energy mode is reached, the command can be reset by the mResetCommand method. This allows commands with lower priority to be executed.

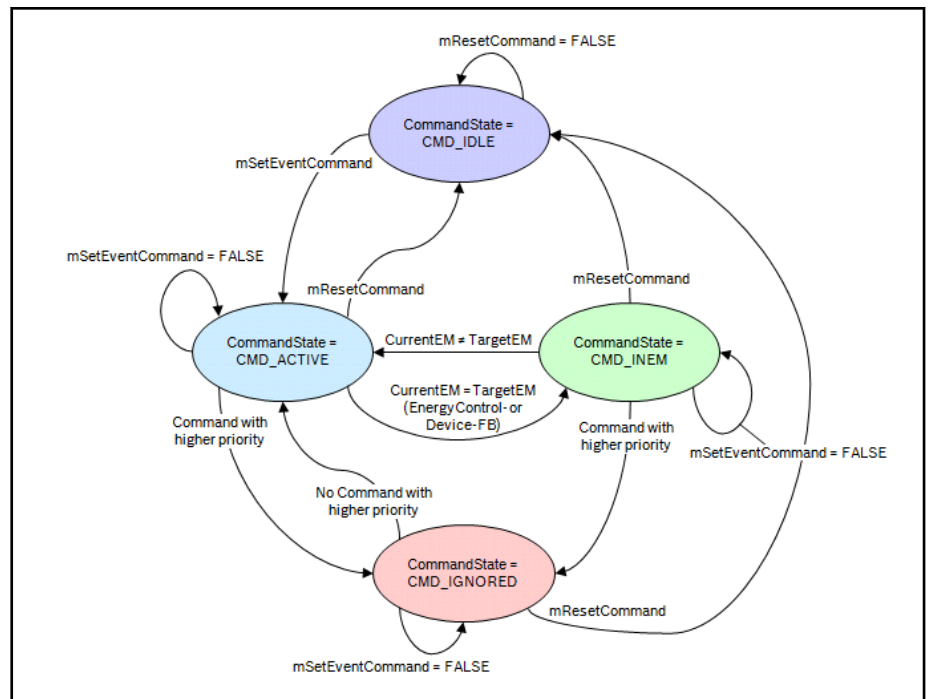


Fig. 9-72: Machine state for commands from IL_4EEEventCommandType02 to the EnergyManagement

The mGetCommandState method can query the status of the command block command.

Sample code:

```

...
(* The variable bChangeEM has to be set by the application if the command
is to be set *)
IF bChangeEM = TRUE THEN
    (* Call of the method mSetEventCommand of the instance fbEventCommand
    in the program IL_4EE *)
    IL_4EE.fbEventCommand.mSetDirectCommand(TargetEM:=> usiTargetEM);
    (* Optional reset the pause condition *)
    bChangeEM := FALSE;
END_IF
...
(* The variable bEndPause has to be set in the Application, if the pause
condition is obsolete *)
IF bEndPause = TRUE THEN
    (* Call of the method mResetCommand of the instance mResetCommand in
    the program IL_4EE, the command gets reset *)
    IL_4EE.fbEventCommand.mResetCommand();
    (* Optional reset the end condition *)
    bEndPause:= FALSE;
END_IF
...
(* By calling the method mGetCommandState the state of the actual command
can be checked *)
IL_4EE.fbEventCommand.mGetCommandState(RemainingTime=> uiRemainingTime, CommandState=> enCmdState)
CASE enCmdState OF
    NO_CMD:
        (* No actual command *)
    CMD_ACTIVE:
        (* The command is in realisation *)
    CMD_INEM:
        (* The command is still actual and has been realised (target energy
        mode has been reached) *)
    CMD_IGNORED:
        (* The command is still actual, but is ignored because of a command
        with higher priority *)
ELSE
    (* Invalid return value *)
END_CASE
...

```

RIL_4EEEnergyManagement.library

END_IF

...



Make sure to also reset a command in the operational energy mode with the mResetCommand method. Otherwise no lower-priority commands can be executed.

Error handling If an error occurs in the block, the commands currently pending are reset. All block methods have no effect and deliver a return value of FALSE.

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A10	16#0001	Cause: The priority cannot be less than 1 Solution: Set priority to a value greater than 0
INPUT_INVALID_ERROR (16#0001)	16#2A11	16#0003	Cause: Higher block does not have TargetEM Solution: Indicate a TargetEM that exists as the EMI-ident for the higher block
OTHER_ERROR (16#7FFE)	16#2A12	16#0001	Cause: Could not register as command block with a higher function block Solution: Adjust NO_OF_COMMANDS_PER_ENERGY_CONTROL or NO_OF_COMMANDS_PER_DEVICE constant in the parameter list
STATE_MACHINE_ERROR (16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-169: Error codes of the IL_4EEEventCommandType02 function blocks

IL_4EEBasicDeviceType01

Brief description

The IL_4EEBasicDeviceType01 function block belongs to the class of device blocks. It serves to specify the energy mode of a consumer. Here the block has to be registered with an EnergyControl block that transmits the current target energy mode to it. Command blocks whose commands should only apply to the corresponding consumer can also be registered.

Make sure that no dead times can be specified for a single IL_4EEBasicDeviceType01 block. This is only indirectly possible through a command block registered with the higher EnergyControl block.

The function block provides the following methods as an interface:

- mGetControl
- mSetState
- mGetState

For details, refer to [Block Methods, page 243](#)

Target system	Library
MTX from 13V06	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-170: Assignment: Target system/library

Interface description

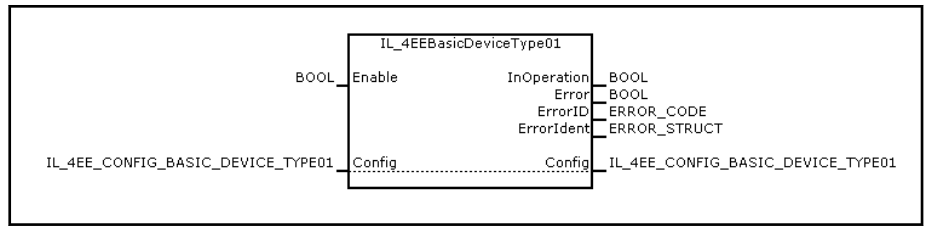


Fig. 9-73: IL_4EEBasicDeviceType01 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	Config	IL_4EE_CONFIG_BASIC_DEVICE_TYPE01	The structure contains the necessary data for configuring the block. IL_4EE_CONFIG_BASIC_DEVICE_TYPE01 , p. 265
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostic, see Error handling , page 247

Tab. 9-171: IL_4EEBasicDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 9-172: Min./max. and default values of the inputs

Block Methods

The function block provides the following methods as an interface:

- [mGetControl](#), p. 243
- [mSetState](#), p. 244
- [mGetState](#), p. 244

mGetControl

The method serves to query the current command from the IL_4EEBasicDeviceType01 function block.

The method should be called cyclically in order to change the energy mode specification.

RIL_4EEEnergyManagement.library

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	Change	IL_4EE_CHANGE_EM			Command to change the energy mode IL_4EE_CHANGE_EM, p. 269
	Target	USINT			EMIdent of target energy mode

Tab. 9-173: Interface for the mGetControl method with min./max. values

mSetState The mSetState method serves to indicate the current energy mode to the IL_4EEBasicDeviceType01 function block. The method only has to be called after implementing a specification queried via mGetControl.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	CurrentEM	USINT			EMIdent of current energy mode

Tab. 9-174: Interface for the mSetState method with min./max. values

mGetState The mGetState method serves to query the current energy mode of the device block.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	CurrentEM	USINT			EMIdent of current energy mode

Tab. 9-175: Interface for the mGetState method with min./max. values

Functional Description

The IL_4EEBasicDeviceType01 block evaluates the commands of the EnergyControl block with which the instance was registered and of the command blocks registered to it.

The command with the highest priority is transmitted to the user interface. The current implementation status is returned to the EnergyControl/command blocks.

mGetControl queries the current command from the application in order to set the energy mode. Once the energy mode is reached, the mSetState method returns the current energy mode to the block. mGetState can, in turn, query this mode.

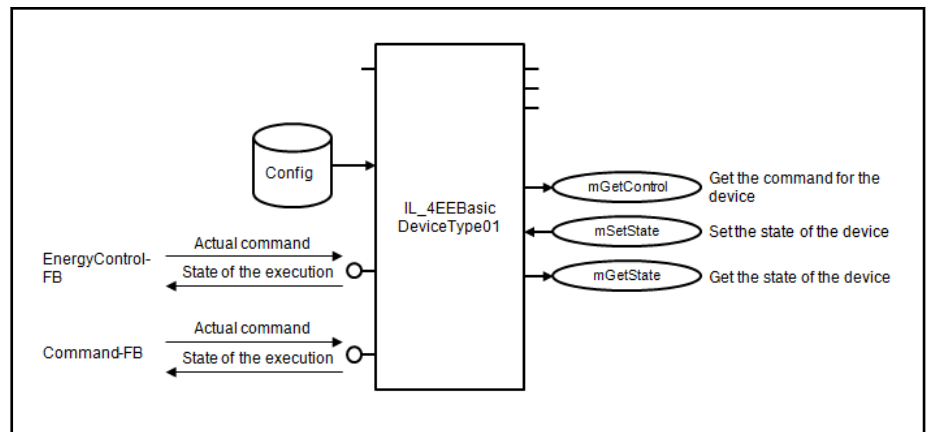


Fig. 9-74: Overview of IL_4EEBasicDeviceType01 block interfaces

Instantiating

The IL_4EEBasicDeviceType01 function block belongs to the class of device blocks. The block has to be registered with an EnergyControl block, which manages the specifications for the target energy mode for the machine or device group. Command blocks whose commands should only apply to the corresponding device can also be registered directly with this block.

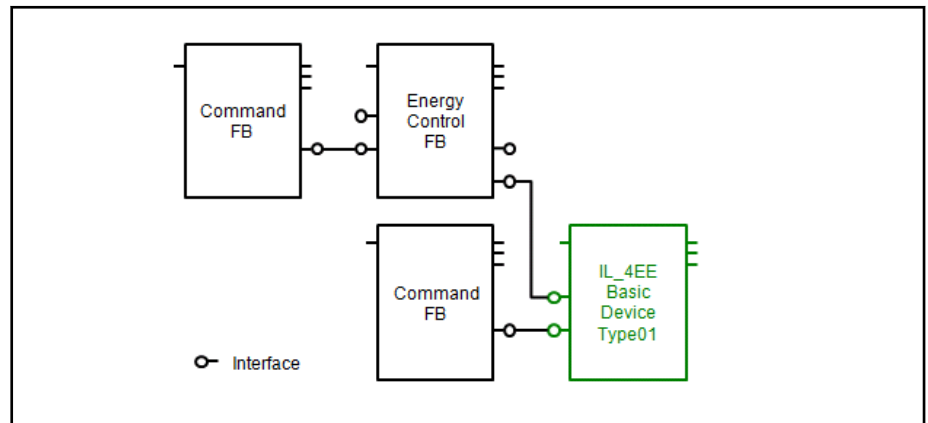


Fig. 9-75: Integrating IL_4EEBasicDeviceType01 into EnergyManagement

The blocks are connected via interfaces.

In order to link the blocks together, the instance of the EnergyControl block with which the instance of the IL_4EEBasicDeviceType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can communicate with the EnergyControl block.

Command blocks can be registered with the instance of the IL_4EEBasicDeviceType01 in the same manner.

SYNTAX

```
VAR
(* Instance of an EnergyControl FB, which is not registered at another FB *)
fbEnergyControl: IL_4EEEnergyControlType01(0);
(* Instance of a Command FB, which is registered at the EnergyControl FB *)
fbMachineEvent: IL_4EEEventCommandType01(fbEnergyControl);
(* Instance of the Device FB, which is registered at the EnergyControl FB *)
fbBasicDevice: IL_4EEBasicDeviceType01(fbEnergyControl);
(* Instance of a Command FB, which is registered at the Device FB *)
fbDeviceEvent: IL_4EEEventCommandType01(fbBasicDevice);
END_VAR
```



The block registration is set by the block declaration and occurs directly after downloading the program to the control system.

It cannot be modified during the program runtime.

RIL_4EEEnergyManagement.library

Maximum Number of Blocks Registering

The number of command blocks that can be registered with an EnergyControl block can be modified with the constant NO_OF_DEVICES_PER_ENERGY_CONTROL in [Parameter list, p. 297](#),. The default value is 32.

The number of command blocks that can be registered with a device block can be modified with the constant NO_OF_COMMANDS_PER_DEVICE in [Parameter list, p. 297](#),. The default value is 4.

Configuration

The block is configured via the input/output structure [IL_4EE_CONFIG_BASIC_DEVICE_TYPE01, p. 265](#).

DeviceName: The name of the device block is needed for the display on the HMI.

PowerConsumptionEMOp: Consumption has to be indicated in the energy mode Operational.



In order to allow for a practical comparison with the other energy modes, the average consumption of the device has to be indicated when the machine is at a standstill in Operational energy mode, not the average consumption while in operation.

The other energy modes are configured in the substructure [IL_4EE_CONFIG_BASIC_DEVICE_EM](#). The following figure shows the definition of an energy mode for the [IL_4EEBasicDeviceType01](#) block.

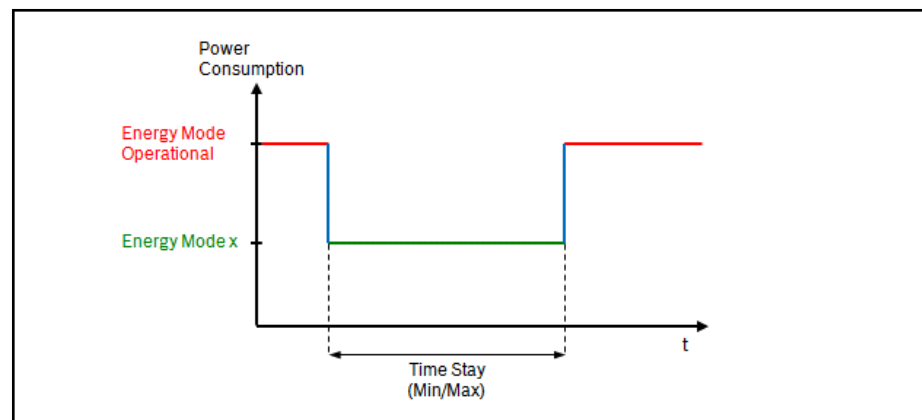


Fig. 9-76: Definition of configuration data for an [IL_4EEBasicDeviceType01](#) energy mode

Configure the following data for each energy mode:

- **EMValid:** The element has to have the value TRUE in order to activate the energy mode configured below
- **EMIdent:** A user-defined, coded hexadecimal value between 16#00 and 16#FE. The EMIdent 16#FF is set by default for the Operational energy mode. The energy mode is output as the target energy mode with this value by the mGetControl method.
- **EMName:** Name of the energy mode for the display on the HMI
- **PowerConsumption:** The average consumption of the device in each energy mode



The number of energy modes of a device block can be modified by the constant NO_OF_EM_PER_DEVICE in the [Parameter list, p. 297](#),. The default value is 4.

EMAssignment: When switching to an energy mode (Direct Mode), a certain energy mode is specified from the application.

Since this is also possible with a higher EnergyControl block, each energy mode for the EnergyControl block with which the instance of IL_4EEBasicDeviceType01 is registered has to be assigned a device energy mode.

Entering the EMIdent in the ControlEM array results in a table assignment:

Array Index x	Energy mode for the higher block EMAssignment[x].ControlEM	Energy mode for the subordinate block EMAssignment[x].TargetEM
0	16#FF	16#FF
1	16#A0	16#FF
2	16#80	16#A0
...		
n	16#20	16#20

Tab. 9-176: Assignment table of energy modes for the instance of IL_4EEBasicDeviceType01 and corresponding instance of IL_4EEEnergyControlType01

Connecting to the Application

The mGetControl method queries the current target energy mode. In addition to the return value, this method has the two outputs, ChangeEM and TargetEM. The outputs are then only valid if the block is in the InOperation state. This is shown by the method return value being TRUE.

If the target energy mode changes, the ChangeEM output has the value CHANGE_ENERGY_MODE (see [IL_4EE_CHANGE_EM](#), p. 269).

The current energy mode is output with the TargetEM output as a coded hexadecimal value according to the configuration.

After implementing the energy mode, it has to be returned in hexadecimal code by the CurrentEM input of the mSetState method.

The ChangeEM output retains the value ChangeEnergyMode until the energy mode is returned by mSetState or the command block specification was aborted.

Sample code:

```

...
(* Call of the method mGetControl, of the instance fbBasicDevice in the program IL_4EE *)
bReturn:= IL_4EE.fbBasicDevice.mGetControl(ChangeEM=> enChangeEM, TargetEM=> usiTargetEM);

IF bReturn = TRUE THEN
  (* The output of the method is valid *)
  IF enChangeEM = ChangeEnergyMode THEN
    (* The energy mode of the output has to be realised *)
    ...
    (* Acknowledgment of the energy mode to the FB fbBasicDevice *)
    IL_4EE.fbBasicDevice.mSetState(CurrentEM:= usiCurrentEM);
  ELSE
    (* No new command *)
  END_IF
ELSE
  (* The outputs of the method are not valid *)
END_IF
...

```

Error handling

If an error occurs, the command to the consumer to switch the energy mode is rescinded (ChangeEM = NO_CHANGE). All block methods have no effect and deliver a return value of FALSE.

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR (16#0006)	16#2A10	16#0000	Cause: A configured EMIdent is greater than 16#FE Solution: Only configure EMIdents less than or equal to 16#FE
INPUT_INVALID_ERROR (16#0001)	16#2A11	16#0000	Cause: EMIdent was used in the configuration more than once Solution: Use each EMIdent only once
INPUT_INVALID_ERROR (16#0001)	16#2A11	16#0001	Cause: Energy mode assigned in EMAssignment, but not configured as EMIdent Solution: Only use configured EMIdent in EMAssignment, check EMValid
INPUT_INVALID_ERROR (16#0001)	16#2A11	16#0002	Cause: The higher EnergyControl block has an energy mode not indicated as ControlEM in Config.EMAssignment Solution: Indicate all EMIdents for higher EnergyControl block as ControlEM
OTHER_ERROR (16#7FFE)	16#2A12	16#0000	Cause: Device block could not be registered with a higher function block Solution: Adjust NO_OF_DEVICES_PER_ENERGY_CONTROL constant in the parameter list
STATE_MACHINE_ERROR (16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-177: Error codes of the IL_4EEBasicDeviceType01 function block

IL_4EEAdvancedDeviceType01

Brief description

The IL_4EEAdvancedDeviceType01 function block belongs to the class of device blocks. It serves to specify the energy mode of a consumer. Here the block has to be registered with an EnergyControl block that transmits the current target energy mode to it. Command blocks whose commands should only apply to the corresponding consumer can also be registered.

In contrast to the IL_4EEBasicDeviceType01 function block, all configuration data are available for IL_4EEAdvancedDeviceType01 after Sercos energy. Thus, pause times can only be specified for this consumer.

The function block provides the following methods as an interface:

- mGetControl
- mSetState
- mGetState

For details, refer to [Block Methods, page 249](#)

Target system	Library
MTX from 14V06	RIL_4EEEnergyManagement
MLC from 14V06	RIL_4EEEnergyManagement

Tab. 9-178: Assignment: Target system/library

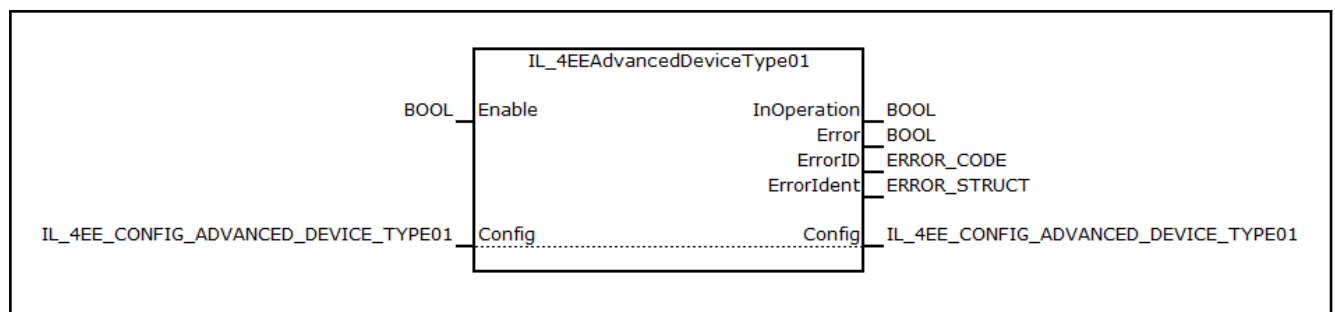


Fig. 9-77: IL_4EEAdvancedDeviceType01 function block

Interface description

I/O type	Name	Data type	Description
VAR_IN_OUT	Config	IL_4EE_CONFIG_ADVANCED_DEVICE_TYPE01	The structure contains the necessary data for configuring the block. IL_4EE_CONFIG_ADVANCED_DEVICE_TYPE01 , p. 266
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostic, see Error handling , page 254

Tab. 9-179: IL_4EEAdvancedDeviceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 9-180: Min./max. and default values of the inputs

Block Methods

The function block provides the following methods as an interface:

- [mGetControl](#), p. 249
- [mSetState](#), p. 250
- [mGetState](#), p. 250

mGetControl

The method serves to query the current command from the IL_4EEAdvancedDeviceType01 function block.

The method should be called cyclically in order to change the energy mode specification.

RIL_4EEEnergyManagement.library

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	Change	IL_4EE_CHANGE_EM			Command to change the energy mode IL_4EE_CHANGE_EM, p. 269
	Target	USINT			EMIdent of target energy mode

Tab. 9-181: Interface for the mGetControl method with min./max. values

mSetState The mSetState method serves to indicate the current energy mode to the IL_4EEAdvancedDeviceType01 function block. The method only has to be called after implementing a specification queried via mGetControl.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	CurrentEM	USINT			EMIdent of current energy mode

Tab. 9-182: Interface for the mSetState method with min./max. values

mGetState The mGetState method is used to query the current energy mode of the IL_4EEAdvancedDeviceType01 function block.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_OUTPUT	CurrentEM	USINT			EMIdent of current energy mode

Tab. 9-183: Interface for the mGetState method with min./max. values

Functional Description

The IL_4EEAdvancedDeviceType01 block evaluates the commands of the EnergyControl block with which the instance was registered and of the command blocks registered to it.

The command with the highest priority is transmitted to the user interface. The current implementation status is returned to the EnergyControl/command blocks.

mGetControl queries the current command from the application in order to set the energy mode. Once the energy mode is reached, the mSetState method returns the current energy mode to the block. mGetState can, in turn, query this mode.

RIL_4EEEnergyManagement.library

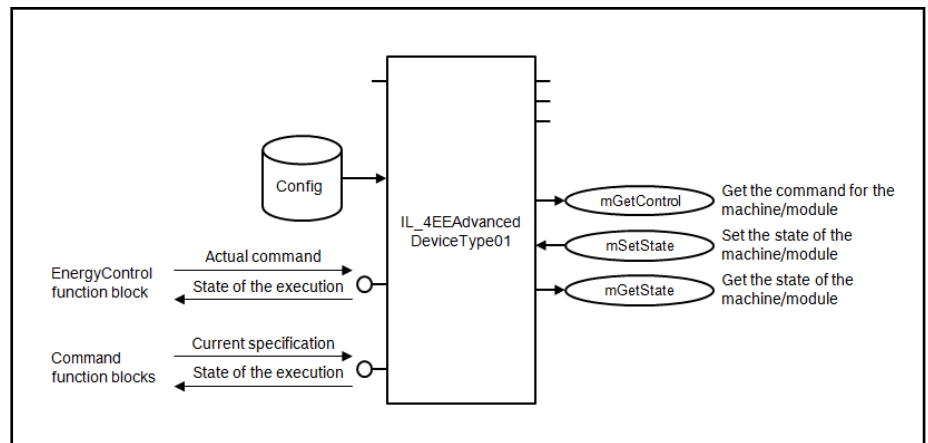


Fig. 9-78: Interfaces of the `IL_4EEAdvancedDeviceType01` function block

Instantiating

The `IL_4EEAdvancedDeviceType01` function block belongs to the class of device blocks. The block has to be registered with an `EnergyControl` block, which manages the specifications for the target energy mode for the machine or device group. Command blocks whose commands should only apply to the corresponding device can also be registered directly with this block.

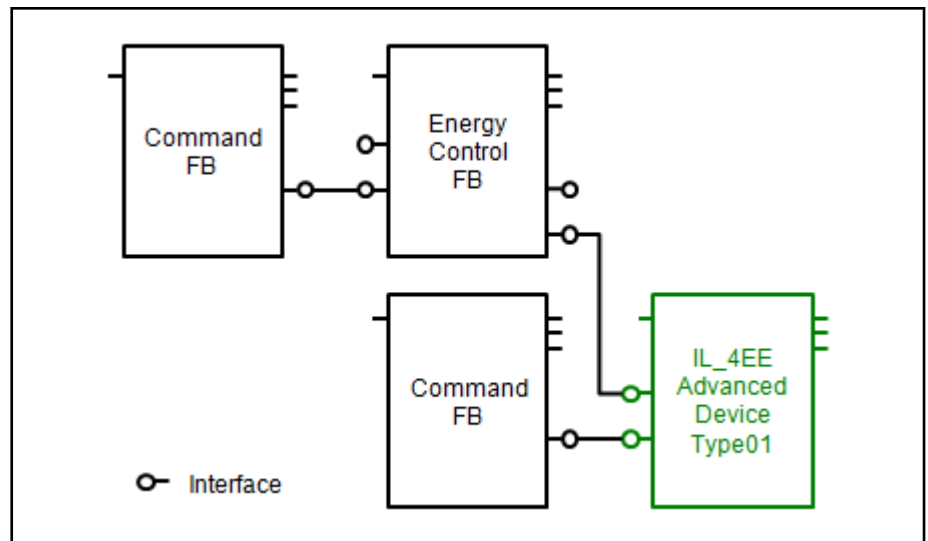


Fig. 9-79: Integrating the `IL_4EEAdvancedDeviceType01` into the `EnergyManagement`

The blocks are connected via interfaces.

In order to link the blocks together, the instance of the `EnergyControl` block with which the instance of the `IL_4EEAdvancedDeviceType01` should be registered has to be specified in the declaration as the input of the `FB_Init` method. With this link, the block can communicate with the `EnergyControl` block.

Command blocks can be registered with the instance of the `IL_4EEAdvancedDeviceType01` in the same manner.

SYNTAX

VAR

```
(* Instance of an EnergyControl function block which does not register at another function block *)
fbEnergyControl: IL_4EEEnergyControlType01(0);
(* Instance of a Command function block which registers at the EnergyControl function block *)
fbMachineEvent: IL_4EEEventCommandType01(fbEnergyControl);
(* Instance of a Device function block which registers at the EnergyControl function block *)
fbAdvancedDevice: IL_4EEAdvancedDeviceType01(fbEnergyControl);
(* Instance of a Command function block which registers at the Device function block *)
```

RIL_4EEEnergyManagement.library

```
fbDeviceEvent: IL_4EEEventCommandType01(fbAdvancedDevice);  
END_VAR
```

 The block registration is set by the block declaration and occurs directly after downloading the program to the control system.
It cannot be modified during the program runtime.

Maximum Number of Blocks Registering

The number of device blocks that can be registered with an EnergyControl block can be modified with the constant "NO_OF_DEVICES_PER_ENERGY_CONTROL" in [Parameter list, page 297](#), The default value is 32.
The number of command blocks that can be registered with an Device block can be modified with the constant "NO_OF_COMMANDS_PER_DEVICE" in [Parameter list, page 297](#), The default value is 4.

Configuration

The block is configured via the input/output structure [IL_4EE_CONFIG_ADVANCED_DEVICE_TYPE01, p. 266](#).

DeviceName: The name of the device block is needed for the display on the HMI.

EMNameOP: Name of the operating energy mode for the display on the HMI.

TimeMinStayEMOp: Minimum dwell time in the operating energy mode. This can be, e.g., used to protect switching contacts. All commands are ignored during this time.

PowerConsumptionEMOp: To calculate an energy mode while specifying a dead time, consumption must be indicated while in the operating energy mode Operational. This mode is generally preset with EMIdent 16#FF.

 In order to allow for a practical comparison with the other energy modes, the average consumption of the device has to be indicated when the machine is at a standstill in operating energy mode, not the average consumption while in operation.

EMConfig: The other energy modes are configured in the substructure [IL_4EE_CONFIG_ADVANCED_DEVICE_EM_TYPE01, p. 267](#),.

The following figure shows the definition of an energy mode. In addition to consumption, the time to transition between the operating energy mode and the energy mode being configured has to be indicated along with the energy needed for the transition.

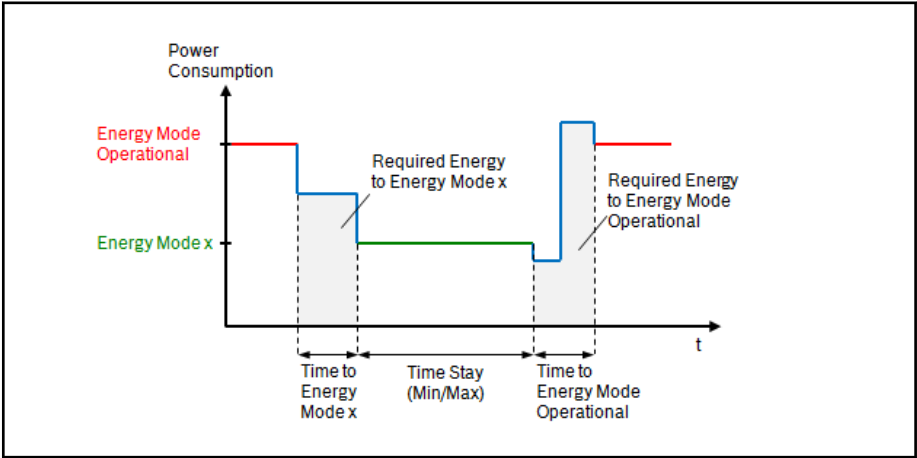


Fig. 9-80: Definition of configuration data for an IL_4EEEnergyControl/Type01 energy mode

Configure the following data for each energy mode:

- **EMValid:** The element has to have the value TRUE in order to activate the energy mode configured below
- **EMIdent:** A user-defined, coded hexadecimal value between 16#00 and 16#FE. The EMIdent 16#FF is set by default for the Operational energy mode. The energy mode is output as the target energy mode with this value by the mGetControl method.
- **EMName:** Name of the energy mode for the display on the HMI
- **TimeToEMx:** Time to transition from the operational energy mode Operational to the energy mode being configured
- **TimeToEMOp:** Time to transition from the energy mode being configured to the operating energy mode Operational
- **TimeMinStay:** Minimum dwell time in the energy mode. This can be, e.g., used to protect switching contacts. All commands are ignored during this time
- **TimeMaxStay:** Maximum dwell time in the energy mode. Once this time elapses, the machine automatically switches to the operating energy mode Operational
- **PowerConsumption:** The average consumption of the device in each energy mode
- **RequEToEMx:** The required energy to transition from the operating energy mode Operational to the energy mode to be configured
- **RequEToEMOp:** The required energy to transition from the operating energy mode Operational to the energy mode to be configured



The number of energy modes of a device block can be modified by the constant NO_OF_EM_PER_DEVICE in the [Parameter list](#), p. 297,.

The default value is 8.

EMAssignment: When switching to an energy mode (Direct Mode), a certain energy mode is specified from the application.

Since this is also possible with a higher EnergyControl block, each energy mode for the EnergyControl block with which the instance of IL_4EEBsaicDeviceType01 is registered has to be assigned a device energy mode.

Entering the EMIdent in the ControlEM array results in a table assignment:

Array Index x	Energy mode for the higher block EMAssignment[x].ControlEM	Energy mode for the subordinate block EMAssignment[x].TargetEM
0	16#FF	16#FF
1	16#A0	16#FF
2	16#80	16#A0
...		
n	16#20	16#20

Tab. 9-184: Assignment table of energy modes for subordinate and higher instance of IL_4EEEnergyControlType01 with cascading

RIL_4EEEnergyManagement.library

Connecting to the Application

The mGetControl method queries the current target energy mode. In addition to the return value, this method has the two outputs, ChangeEM and TargetEM. The outputs are then only valid if the block is in the InOperation state. This is shown by the method return value being TRUE.

If the target energy mode changes, the ChangeEM output has the value CHANGE_ENERGY_MODE (see [IL_4EE_CHANGE_EM](#), page 269).

The current energy mode is output with the TargetEM output as a coded hexadecimal value according to the configuration.

After implementing the energy mode, it has to be returned in hexadecimal code by the CurrentEM input of the mSetState method.

The ChangeEM output retains the value ChangeEnergyMode until the energy mode is returned by mSetState or the command block specification was aborted.

Sample code:

```
...
(* Calling the mGetControl method of the fbAdvancedDevice instance in the IL 4EE program *)
bReturn:= IL_4EE.fbAdvancedDevice.mGetControl(ChangeEM=> enChangeEM, TargetEM=> usiTargetEM);

IF bReturn = TRUE THEN
  (* The method outputs are valid *)
  IF enChangeEM = ChangeEnergyMode THEN
    (* The energy mode of the usiTargetEM output has to be established*)
    ...
    (* Energy mode feedback to the fbAdvancedDevice instance *)
    IL_4EE.fbAdvancedDevice.mSetState(CurrentEM:= usiCurrentEM);
  ELSE
    (* The energy mode of the consumer is up-to-date *)
  END_IF
ELSE
  (* The method outputs are invalid *)
END_IF
...
```

Error handling

If an error occurs, the command to the consumer to switch the energy mode is canceled (ChangeEM = NO_CHANGE). All block methods have no effect and deliver a return value of FALSE.

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
INPUT_RANGE_ERROR	16#2A10	16#0000	Cause: A configured EMIdent is greater than 16#FE Solution: Only configure EMIdents less than or equal to 16#FE
INPUT_INVALID_ERROR	16#2A11	16#0000	Cause: EMIdent was used in the configuration more than once Solution: Use each EMIdent only once
INPUT_INVALID_ERROR	16#2A11	16#0001	Cause: Energy mode assigned in EMAssignment, but not configured as EMIdent Solution: Only use configured EMIdent in EMAssignment, check EMValid
INPUT_INVALID_ERROR	16#2A11	16#0002	Cause: The higher EnergyControl block has an energy mode not indicated as ControlEM in Config.EMAssignment Solution: Indicate all EMIdents for higher EnergyControl block as ControlEM

ErrorID	Additional1	Additional2	Description
OTHER_ERROR	16#2A12	16#0000	Cause: Device block could not be registered with a higher function block Solution: Adjust NO_OF_DEVICES_PER_ENERGY_CONTROL constant in the parameter list
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-185: Error codes of the IL_4EEAdvancedDeviceType01 function block

IL_4EEEnergyFunctionType01

Brief description The IL_4EEEnergyFunctionType01 function block belongs to the class of EnergyFunction blocks. It serves to visualize analogous values from existing energy saving functions (e.g., power percentage value of a fan).

Here the block is registered with an EnergyControl block.

The data is visualized from one HMIDataCollector block to this registered one.

The function block provides the following method as an interface:

- mSetValue

For details, refer to [Block Methods, page 256](#)

Target system	Library
MTX from 13V06	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-186: Assignment: Target system/library

Interface description

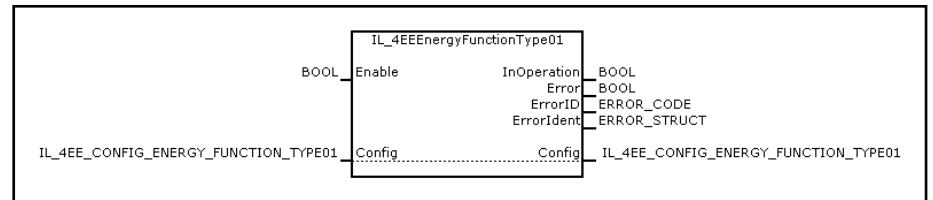


Fig. 9-81: IL_4EEEnergyFunctionType01 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	Config	IL_4EE_CONFIG_ENERGY_FUNCTION_TYPE01	The structure contains the necessary data for configuring the block. See IL_4EE_CONFIG_ENERGY_FUNCTION_TYPE01, page 268
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Error handling, page 257

Tab. 9-187: IL_4EEEnergyFunctionType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Block Methods The function block provides the following methods as an interface:

- [mSetValue, p. 256](#)

mSetValue The method serves to transmit the current energy saving function value to the block.

I/O type	Name	Type	Min. value	Max. value	Description
RETURN_VALUE		BOOL			TRUE: Method executed successfully FALSE: Method not executed successfully and no valid values returned
VAR_INPUT	Value	REAL			Command to change the energy mode

Tab. 9-188: Interface for the mSetValue method with min./max. values

Functional Description

The IL_4EEEnergyFunctionType01 block receives the current consumption value of the corresponding device from the mSetValue method. This is read by an HMIDataCollector block that is also registered with an EnergyControl block and provided to the HMI structure for visualization.

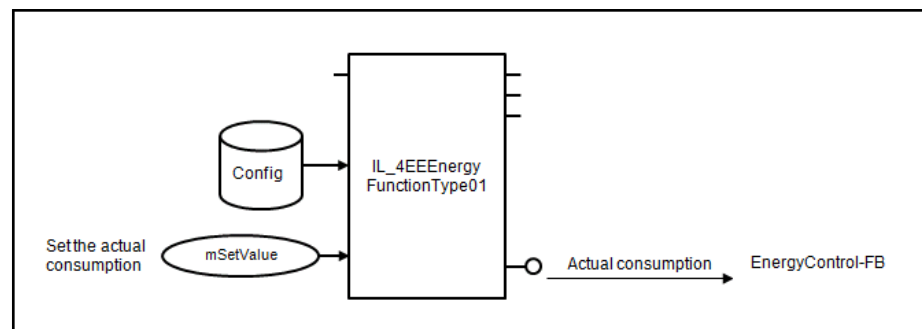


Fig. 9-82: Overview of IL_4EEEnergyFunctionType01 block interfaces

Instantiating

The IL_4EEEnergyFunctionType01 function block belongs to the class of EnergyFunction blocks. The block has to be registered with an EnergyControl block. An HMIDataCollector registered with this provides the consumption value for the energy saving function to the HMI.

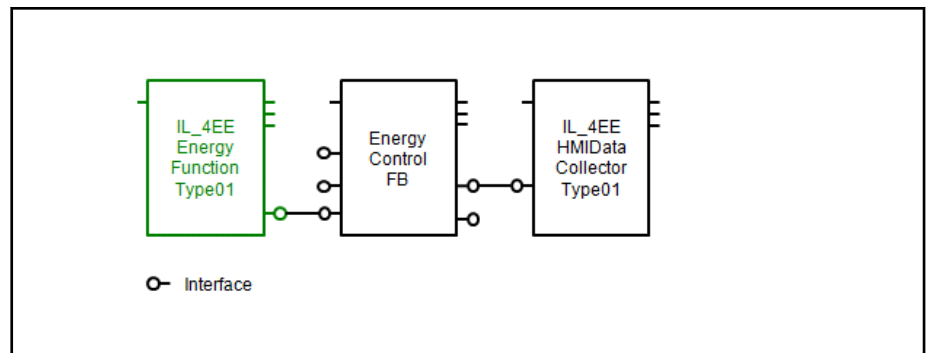


Fig. 9-83: Integrating IL_4EEEnergyFunctionType01 into EnergyManagement
The blocks are connected via interfaces.

In order to link the blocks together, the instance of the EnergyControl block with which the instance of the IL_4EEEnergyFunctionType01 should be registered has to be specified in the declaration as the input of the FB_Init method. With this link, the block can communicate with the EnergyControl block.

An HMIDataCollector block can be registered with the instance of the EnergyControl block in the same manner.

SYNTAX

```
VAR
(* Instance of the EnergyControl FB, which doesn't register itself *)
fbEnergyControl: IL_4EEEnergyControlType01(0);
(* Instance of a EnergyFunction FB, which registers itself at the
EnergyControl FB *)
fbEnergyFunction: IL_4EEEnergyFunctionType01(fbEnergyControl);
(* Instance of a HMIDataCollector FB, which registers itself at the
EnergyControl FB *)
fbHMIDataCollector: IL_4EEHMIDataCollectorType01(fbEnergyControl);
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

Maximum Number of Blocks Registering

The number of EnergyFunction blocks that can be registered with an EnergyControl block can be modified with the constant NO_OF_ENERGY_FUNCTIONS_PER_HMI in [Parameter list, p. 297](#). The default value is 0.

Error handling

If an error occurs, all block methods have no effect and return the value FALSE.

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
OTHER_ERROR (16#7FFE)	16#2A12	16#0002	Cause: Could not register as EnergyFunction block with a higher function block Solution: Adjust NO_OF_ENERGY_FUNCTIONS_PER_HMI constant in the parameter list
STATE_MACHINE_ERROR (16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-189: Error codes of the IL_4EEEnergyFunctionType01 function block

IL_4EEHMIDataCollectorType01

Brief description

The IL_4EEHMIDataCollectorType01 function block serves to gather the data needed for the HMI controls.

RIL_4EEEnergyManagement.library

Here the block is registered with an EnergyControl block. The block then gathers the data from all device, command and EnergyControl blocks registered under the same EnergyControl block.

The data is then saved in a structure ([IL_4EE_HMI_DATA_TYPE01](#), p. 269,) specified for the HMI controls.

Target system	Library
MTX from 13V06	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-190: Assignment: Target system/library

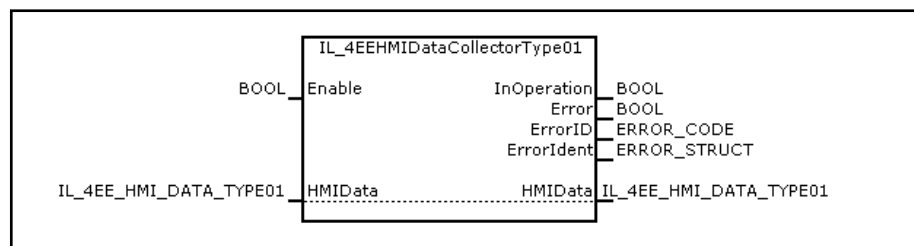
Interface description

Fig. 9-84: IL_4EEHMIDataCollectorType01 function block

I/O type	Name	Data type	Description
VAR_IN_OUT	HMIData	IL_4EE_HMI_DATA_TYPE01	Data needed by the HMI controls See IL_4EE_HMI_DATA_TYPE01 , page 269
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCT	Detailed diagnostic, see Error handling , page 259

Tab. 9-191: IL_4EEHMIDataCollectorType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous

Tab. 9-192: Minimum/maximum values and default values of the IL_4EEHMIDataCollector inputs

Functional Description**Instantiating**

The IL_4EEHMIDataCollectorType01 function block serves to gather the data needed by the HMI controls.

This only applies to the data for controlling the devices, not for monitoring.

The block has to be registered with an EnergyControl block. This transmits the interfaces for all device, command and EnergyControl blocks registered to it to the HMIDataCollector block. This allows the HMIDataCollector block to access all blocks subordinate to the EnergyControl block.

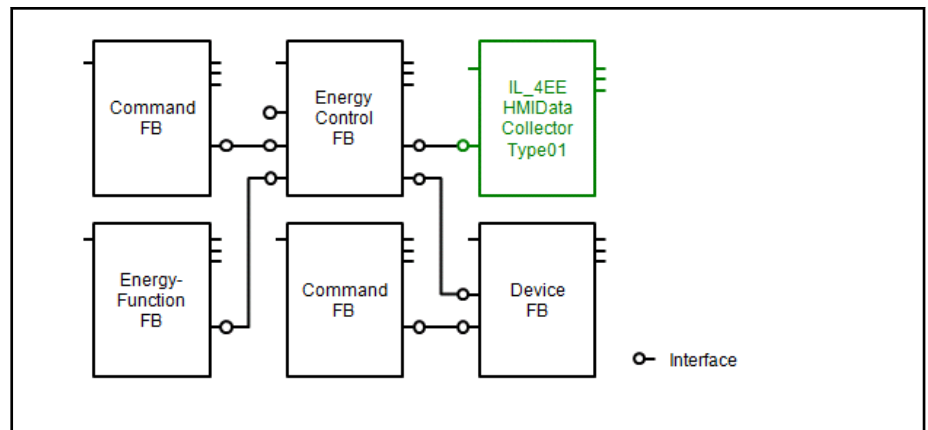


Fig. 9-85: Integrating IL_4EEHMIDataCollectorType01 into EnergyManagement

The blocks are connected via interfaces.

In order to link the blocks together, the instance of the EnergyControl block with which the instance of the IL_4EEHMIDataCollectorType01 should be registered has to be specified in the declaration as the input of the FB_Init method. The block registers with this link.

SYNTAX

```
VAR
(* Instance of an EnergyControl FB, which doesn't register itself *)
fbEnergyControl: IL_4EEEnergyControlType01(0);
(* Instance of a HMIDDataCollector FB, which registers itself at the
EnergyControl FB *)
fbHMIDDataCollector: IL_4EEHMIDDataCollectorType01(fbEnergyControl);
(* Instance of a Command FB, which registers itself at the
EnergyControl FB *)
fbBasicCommand: IL_4EEBasicCommandType01(fbEnergyControl);
(* Instance of a Device FB, which registers itself at the
EnergyControl FB *)
fbBasicDevice: IL_4EEExtendedDeviceType01(fbEnergyControl);
(* Instance of a Command FB, which registers itself at the Device FB*)
fbDeviceEvent: IL_4EEEventCommandType01(fbBasicDevice);
END_VAR
```



The block registration is set by the block declaration and occurs directly after downloading the program to the control system.

It cannot be modified during the program runtime.

Call

The HMIDDataCollector block cannot be integrated into the application like other blocks. The block only has to be enabled via the Enable input and called cyclically.

Error handling

The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
OTHER_ERROR (16#7FFE)	16#2A12	16#0003	Could not register as HMIDDataCollector block with a higher function block
STATE_MACHINE_ERROR (16#0005)	16#0006	16#FFFF	Invalid function block state

Tab. 9-193: Error codes of the IL_4EEHMIDDataCollectorType01 function block

IL_4EEScadaDataCollectorType01

Brief description

The function blocks connects the EnergyControl on the control to a control system/MES. The function block is registered at a EnergyControl function

RIL_4EEEnergyManagement.library

block which represents the machine. The communication with the control system is realized via global structures that have to be applied to the function block. Via these structures, the configuration and status data of the Energy-Control are specified. Specifications can also be transmitted to the Energy-Control via these structures.

Target system	Library
MTX from 14V06	RIL_4EEEnergyManagement
MLC from 14V06	RIL_4EEEnergyManagement

Tab. 9-194: Assignment: Target system/library

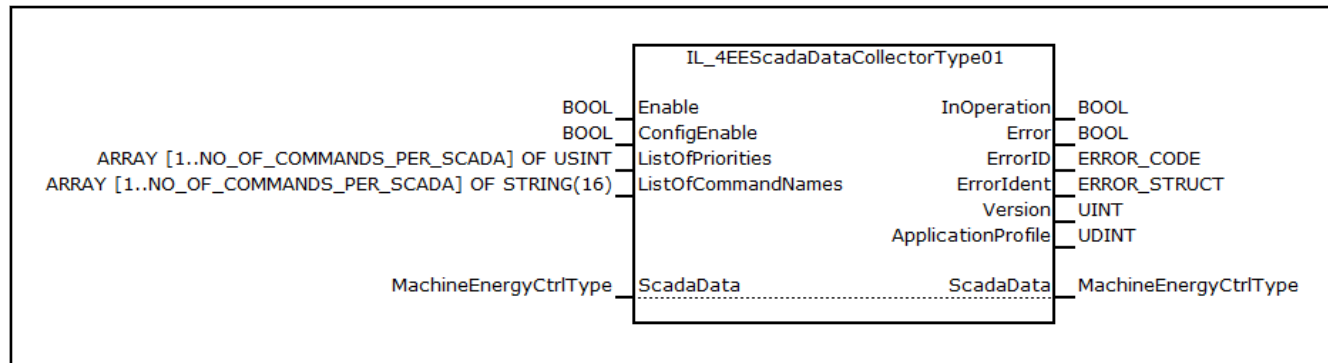


Fig. 9-86: IL_4EEScadaDataCollectorType01 function block

Interface description

I/O type	Name	Data type	Description
VAR_IN_OUT	ScadaData	MachineEnergyCtrlType	Structure with status, configuration and default data for communication with the control system/MES MachineEnergyCtrlType, p. 293
VAR_INPUT	Enable	BOOL	Processing of function block enabled (level-controlled)
	ConfigEnable	BOOL	Enabling the generation of the energy mode properties of the machine (MachineEnergyCtrlConfig structure)
	ListOfPriorities	ARRAY [1..NO_OF_COMMANDS_PER_SCADA] OF USINT	Priorities of the specifications to the control system/MES for the Energy-Control function block in the order of the specifications in the Scada data structure
	ListOfCommandNames	ARRAY [1..NO_OF_COMMANDS_PER_SCADA] OF STRING(16)	Names of the specifications to the control system/MES for the Energy-Control function block in the order of the specifications in the Scada data structure
VAR_OUTPUT	InOperation	BOOL	Function block is operating
	Error	BOOL	Processing completed with error

I/O type	Name	Data type	Description
	ErrorID	ERROR_CODE	Diagnostic description in the event of an error
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Error handling, page 263
	Version	UINT	Displays the interface version (0x0001)
	Application-Profile	UDINT	Displays the enabled application profile (is OR-ed with the IL_4EEMon-ScadaDataType01 display and to be written to the structure)

Tab. 9-195: IL_4EEMonScadaDataCollectorType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Enable	BOOL			FALSE	Continuous
ConfigEnable	BOOL			FALSE	Pos. edge at Enable
ListOfPriorities	ARRAY [1..NO_OF_COMMANDS_PER_SCADA] OF USINT	1	255	127	Pos. edge at Enable
ListOfCommand-Names	ARRAY [1..NO_OF_COMMANDS_PER_SCADA] OF STRING(16)				Pos. edge at Enable

Tab. 9-196: Min./max. and default values of the inputs

Functional Description

The IL_4EEScadaDataCollectorType01 function block prepares the EnergyControl data required by the control system/MES. This only applies to the data for controlling the devices, not for monitoring. Furthermore, the function block manages specifications sent from the control system/MES to the EnergyControl function block. Thus, the function block is also used as Command block. The function block is registered at a EnergyControl function block which represents the machine.

RIL_4EEEnergyManagement.library

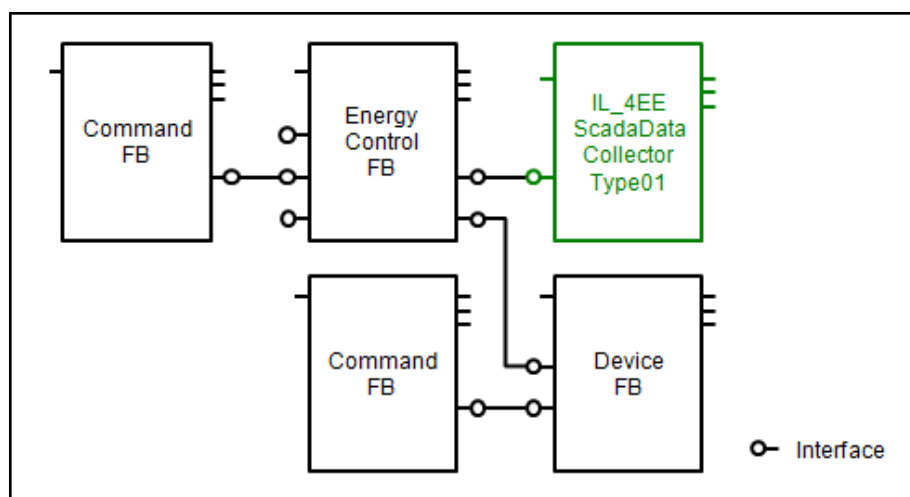


Fig. 9-87: Integrating the IL_4EE ScadaDataCollectorType01 into Energy Management

The blocks are connected via interfaces. In order to link the blocks together, the instance of the EnergyControl block with which the instance of the IL_4EE ScadaDataCollectorType01 should be registered has to be specified in the declaration as the input of the FB_Init method. The block registers with this link.

SYNTAX

```
VAR
(* Instance of an EnergyControl function block which does not register at another function block *)
fbEnergyControl: IL_4EEEnergyControlType01(0);
(* Instance of a ScadaDataCollector function block which registers at the EnergyControl function block *)
fbScadaDataCollector: IL_4EEScadaDataCollectorType01(fbEnergyControl);
(* Instance of a Command function block which registers at the EnergyControl function block *)
fbMachineEvent: IL_4EEEventCommandType01(fbEnergyControl);
(* Instance of a Device function block which registers at the EnergyControl function block *)
fbAdvancedDevice: IL_4EEAdvancedDeviceType01(fbEnergyControl);
(* Instance of a Command function block which registers at the Device function block *)
fbDeviceEvent: IL_4EEEventCommandType01(fbAdvancedDevice);
END_VAR
```

The block registration is set by the block declaration and occurs directly after downloading the program to the control system. It cannot be modified during the program runtime.

- Call** The ScadaDataCollector function block cannot be integrated into the application as other blocks. The block only has to be enabled via the Enable input and called cyclically.
- Function** The IL_4EE ScadaDataCollectorType01 reads and writes a structure of type MachineEnergyCtrlType (substructure of MachineEnergyObjectType).

The structure contains the four substructures:

- **MachineEnergyCtrlConfig:**
The structure contains the configuration data of the energy modes of the machine (of the EnergyControl function block). The ConfigEnable input has to enable writing of this substructure.
- **MachineEnergyCtrlStatus:**
The structure contains the current status data of the machine (of the EnergyControl function block). This substructure is always written by the function block.
- **MachineEnergyCtrlCmdControlType:**

RIL_4EEEnergyManagement.library

The structure contains the specifications sent from the control system/MES to the EnergyControl function block; i.e., it is written by the control system/MES. The Command function blocks are contained in the function block. Function block priorities as well as their name on the HMI can be configured via the input arrays ListOfPriority and ListOfCommandNames.

- MachineEnergyCtrlCmdStatus:

The structure contains the specification status of the control system/MES.

Error handling The function block uses the error table **F_RELATED_TABLE, 16#0170**. It can generate the following error messages in Additional1/Additional2:

ErrorID	Additional1	Additional2	Description
OTHER_ERROR	16#2A12	16#0003	Cause: Could not register as HMIDataCollector block with a higher function block Solution: Specify EnergyControl block in the declaration
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-197: Error codes of the IL_4EEscadaDataCollectorType01 function block

9.3.3 Data types

IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01

Brief Description The structure IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01 contains the necessary data for configuring the IL_4EEEnergyControlType01 block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
ControlName	STRING(16)			" "	Pos. edge "Enable"	Name of the EnergyControl block (machine or module)
EMNameOp	STRING(16)			"Operational"	Pos. edge "Enable"	Name of the operational energy mode
PowerConsumptionEMOp	REAL			0	Pos. edge "Enable"	Average energy consumption in [W] in the energy mode Operational
TimeMinStayEMOp	TIME				Pos. edge "Enable"	Minimum dwell time in the operational energy mode
EMConfig	ARRAY [1..NO_OF_EM_PER_ENERGY_CONTROL] OF IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01				Pos. edge "Enable"	Array of structures for configuring the energy modes Parameter list, p. 297, IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01, p. 264

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
EMAssignment	ARRAY [0..NO_OF_EM_PER_ENERGY_CONTROL] OF IL_4EE_EM_ASSIGNMENT_TYPE01				Pos. edge "Enable"	Assignment of the EM for the block to the EM of the higher EnergyControl function block (only needed for cascading) Parameter list, p. 297, IL_4EE_EM_ASSIGNMENT_TYPE01, p. 268
StatusOverride	BOOL			FALSE	Pos. edge "Enable"	TRUE: The mSetState method returns the current energy mode FALSE: The current energy mode is determined from the energy modes of the devices

Tab. 9-198: Min./max. and default values of the IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01 structure

IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01

Brief Description

The IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01 substructure for the structure [IL_4EE_CONFIG_ENERGY_CONTROL_TYPE01, p. 263](#), contains the data needed for setting the energy modes for configuring the [IL_4EEEnergyControlType01, p. 212](#) block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
EMValid	BOOL			FALSE	Pos. edge "Enable"	TRUE: The energy mode has been configured FALSE: The energy mode is not included
EMIdent	USINT		16#FE	16#00	Pos. edge "Enable"	ID of the energy mode (hexadecimal code)
EMName	STRING(16)			" "	Pos. edge "Enable"	Name of the energy mode, e.g., "StandBy"
TimeToEMx	TIME			0	Pos. edge "Enable"	Time needed to transition from Operational to energy mode x
TimeToEMOp	TIME			0	Pos. edge "Enable"	Time [s] needed to transition from energy mode x to Operational
TimeMinStay	TIME			0	Pos. edge "Enable"	Minimum time [s] that has to be spent in the energy mode
TimeMaxStay	TIME			0	Pos. edge "Enable"	Maximum time [s] that can be spent in the energy mode; if the variable has a value of 0, it is ignored
PowerConsumption	REAL			0	Pos. edge "Enable"	Average energy consumption in [W] in energy mode x

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
RequEToEMx	REAL			0	Pos. edge "Enable"	Energy consumption in [W] to switch from Operational to energy mode x
RequEToEMOp	REAL			0	Pos. edge "Enable"	Energy consumption in [W] to switch from energy mode x to Operational

Tab. 9-199: Min./max. and default values of the IL_4EE_CONFIG_ENERGY_CONTROL_EM_TYPE01 structure

IL_4EE_CONFIG_EVENT_COMMAND_TYPE01

Brief Description The structure contains the necessary data for configuring the [IL_4EEEventCommandType01](#), p. 228 block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
CommandName	STRING(16)			" "	Pos. edge "Enable"	Name of the command
TargetEM	USINT				Pos. edge "Enable"	Target energy mode for the command (value range: 16#00 - 16#FF)
Priority	USINT	1		127	Pos. edge "Enable"	Command priority (1: highest priority; 255: lowest priority)
DelayTime	TIME			0	Pos. edge "Enable"	Time delay [s] until command becomes active

Tab. 9-200: Min./max. and default values of the IL_4EE_CONFIG_EVENT_COMMAND_TYPE01 structure

IL_4EE_CONFIG_BASIC_DEVICE_TYPE01

The structure contains the necessary data for configuring the [IL_4EEBasicDeviceType01](#), p. 242 block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
DeviceName	STRING(16)			" "	Pos. edge "Enable"	Name of the device
EMNameOp	STRING(16)			"Operational"	Pos. edge "Enable"	Name of the operational energy mode
PowerConsumptionEMOp	REAL			0	Pos. edge "Enable"	Average energy consumption in [W] in the energy mode Operational

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
EMConfig	ARRAY [1..NO_OF_EM_PER_DEVICE] OF IL_4EE_CONFIG_BASIC_DEVICE_EM_TYPE01				Pos. edge "Enable"	Array of structures for configuring the energy modes see Parameter list , p. 297, IL_4EE_CONFIG_BASIC_DEVICE_EM_TYPE01 , p. 266,
EMAssignment	ARRAY [1..NO_OF_EM_PER_ENERGY_CONTROL] OF IL_4EE_EM_ASSIGNMENT_TYPE01				Pos. edge "Enable"	Assignment of the EM for the device to the EM of the higher EnergyControl function block see Parameter list , p. 297, IL_4EE_EM_ASSIGNMENT_TYPE01 , p. 268

Tab. 9-201: Min./max. and default values of the IL_4EE_CONFIG_BASIC_DEVICE_EM_TYPE01 structure

IL_4EE_CONFIG_BASIC_DEVICE_EM_TYPE01

Brief Description The IL_4EE_CONFIG_BASIC_DEVICE_EM_TYPE01 substructure for the [IL_4EE_CONFIG_BASIC_DEVICE_TYPE01](#), p. 265, structure contains the configuration for an energy mode for the [IL_4EEBasicDeviceType01](#), p. 242 block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
EMValid	BOOL			FALSE	Pos. edge "Enable"	TRUE: Energy mode included by block
EMIdent	USINT		16#FE	16#00	Pos. edge "Enable"	ID of the energy mode (hexadecimal code)
EMName	STRING(16)			" "	Pos. edge "Enable"	Name of the energy mode, e.g., "Operational"
PowerConsumption	REAL			0	Pos. edge "Enable"	Average energy consumption in [W] in the energy mode

Tab. 9-202: Min./max. and default values of the IL_4EE_CONFIG_BASIC_DEVICE_EM_TYPE01 structure

IL_4EE_CONFIG_ADVANCED_DEVICE_TYPE01

Brief description The IL_4EE_CONFIG_ADVANCED_DEVICE_TYPE01 structure contains the data required to configure the function block [IL_4EEAdvancedDeviceType01](#), page 248.

Name	Type	Min. value	Max. value	Default value	Effective	Description
DeviceName	STRING(16)			" "	Pos. edge "Enable"	Name of the Device function block (consumer)
EMNameOp	STRING(16)			"Operational"	Pos. edge "Enable"	Name of the operational energy mode

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
PowerConsumptionEMOp	REAL			0	Pos. edge "Enable"	Average energy consumption in [W] in the energy mode Operational
TimeMinStayEMOp	TIME			T#0S	Pos. edge "Enable"	Minimum dwell time in the operational energy mode
EMConfig	ARRAY [1..NO_OF_EM_PER_DEVICE] OF IL_4EE_CONFIG_ADVANCED_DEVICE_EM_TYPE01				Pos. edge "Enable"	Array of structures for configuring the energy modes Parameter list, page 297, IL_4EE_CONFIG_ADVANCED_DEVICE_EM_TYPE01, p. 267
EMAssignment	ARRAY [0..NO_OF_EM_PER_ENERGY_CONTROL] OF IL_4EE_EM_ASSIGNMENT_TYPE01				Pos. edge "Enable"	Assignment of the EM of the function block to the EM of the higher EnergyControl function block Parameter list, page 297, IL_4EE_EM_ASSIGNMENT_TYPE01, p. 268

Tab. 9-203: Min./max. and default values of the IL_4EE_CONFIG_ADVANCED_DEVICE_TYPE01 structure

IL_4EE_CONFIG_ADVANCED_DEVICE_EM_TYPE01

Brief description The IL_4EE_CONFIG_ADVANCED_DEVICE_TYPE01 substructure for the structure [IL_4EE_CONFIG_ADVANCED_DEVICE_TYPE01, p. 266](#), contains the data needed for setting the energy modes for configuring the [IL_4EEAdvancedDeviceType01, p. 248](#) block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
EMValid	BOOL			FALSE	Pos. edge "Enable"	TRUE: The energy mode has been configured FALSE: The energy mode is not included
EMIdent	USINT		16#FE	16#00	Pos. edge "Enable"	ID of the energy mode (hexadecimal code)
EMName	STRING(16)			""	Pos. edge "Enable"	Name of the energy mode, e.g., "Stand-By"
TimeToEMx	TIME			T#0S	Pos. edge "Enable"	Time needed to transition from Operational to energy mode x
TimeToEMOp	TIME			T#0S	Pos. edge "Enable"	Time [s] needed to transition from energy mode x to Operational
TimeMinStay	TIME			T#0S	Pos. edge "Enable"	Minimum time [s] that has to be spend in the energy mode

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
TimeMaxStay	TIME			T#0S	Pos. edge "Enable"	Maximum time [s] that can be spent in the energy mode; if the variable has a value of 0, it is ignored
PowerConsumption	REAL			0	Pos. edge "Enable"	Average energy consumption in [W] in energy mode x
RequEToEMx	REAL			0	Pos. edge "Enable"	Energy consumption in [W] to switch from Operational to energy mode x
RequEToEMOp	REAL			0	Pos. edge "Enable"	Energy consumption in [W] to switch from energy mode x to Operational

Tab. 9-204: Min./max. and default values of the IL_4EE_CONFIG_ADVANCED_DEVICE_EM_TYPE01 structure

IL_4EE_CONFIG_BASIC_COMMAND_TYPE01

Brief Description The structure describes the configuration data for the block [IL_4EEBasic-CommandType01](#), p. 221.

Name	Type	Min. value	Max. value	Default value	Effective	Description
CommandName	STRING(16)			" "	Pos. edge "Enable"	Name of the command
Priority	USINT	1		127	Pos. edge "Enable"	Command priority (1: highest priority; 255: lowest priority)

Tab. 9-205: Min./max. and default values of the IL_4EE_CONFIG_BASIC_COMMAND structure

IL_4EE_CONFIG_ENERGY_FUNCTION_TYPE01

Brief Description The IL_4EE_CONFIG_ENERGY_FUNCTION_TYPE01 structure contains the configuration data for the block [IL_4EEEnergyFunctionType01](#), p. 255.

Name	Type	Min. value	Max. value	Default value	Effective	Description
FunctionName	STRING(16)			" "	Pos. edge "Enable"	Name of the energy saving function
Deviceldx	USINT			0	Pos. edge "Enable"	Name of the operational energy mode

Tab. 9-206: Min./max. and default values of the IL_4EE_CONFIG_ENERGY_FUNCTION_TYPE01 structure

IL_4EE_EM_ASSIGNMENT_TYPE01

Brief Description The IL_4EE_EM_ASSIGNMENT_TYPE01 structure contains the data necessary for assigning the energy modes (EM) of a block to the EM of the higher EnergyControl function block (only needed for cascading).

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
ControlEM	USINT			16#FF	Pos. edge "Enable"	ID of the target energy mode of the higher EnergyControl block (hexadecimal code)
TargetEM	USINT			16#FF	Pos. edge "Enable"	ID of the assigned target energy mode of the subordinate block (hexadecimal code)

Tab. 9-207: Min./max. and default values of the IL_4EE_EM_ASSIGNMENT_TYPE01 structure

IL_4EE_TYPE_SEL

Brief Description The IL_4EE_TYPE_SEL enumeration serves to select a command type.

Element	Value	Description
DIRECT_MODE	0	Direct specification of a target energy mode
PAUSE_TIME_MODE	1	Dead time is specified. The target energy mode is calculated. Switching to the operational energy mode requires a separate command
PAUSE_TIME_WAKE_UP_MODE	2	Dead time is specified. The target energy mode is calculated. Switching to the operational energy mode is automatic after dead time elapses

Tab. 9-208: Elements of the IL_4EE_TYPE_SEL enumeration type

IL_4EE_CHANGE_EM

Brief Description The IL_4EE_CHANGE_EM enumeration is used in the mGetControl method of the EnergyControl and device blocks in order to show a change in the target energy mode.

Element	Value	Description
NO_CHANGE	0	No change to energy mode
CHANGE_ENERGY_MODE	1	Change energy mode

Tab. 9-209: Elements of the IL_4EE_CHANGE_EM enumeration type

IL_4EE_CMD_STATE

Brief Description The IL_4EE_CMD_STATE enumeration serves to display the status of a command block on the HMI.

Element	Value	Description
CMD_IDLE	0	No command recognized
CMD_ACTIVE	1	Current command is being executed
CMD_INEM	2	Current command has been executed and is still pending
CMD_IGNORED	3	The current command is ignored due to a higher-priority command

Tab. 9-210: Elements of the IL_4EE_CMD_STATE enumeration type

9.3.4 HMI control data types

IL_4EE_HMI_DATA_TYPE01

Brief Description The IL_4EE_HMI_DATA_TYPE01 structure contains the data for the HMI controls.

RIL_4EEEnergyManagement.library

These are written by the block [IL_4EEHMIDataCollectorType01](#), p. 257, and do not have to be set.

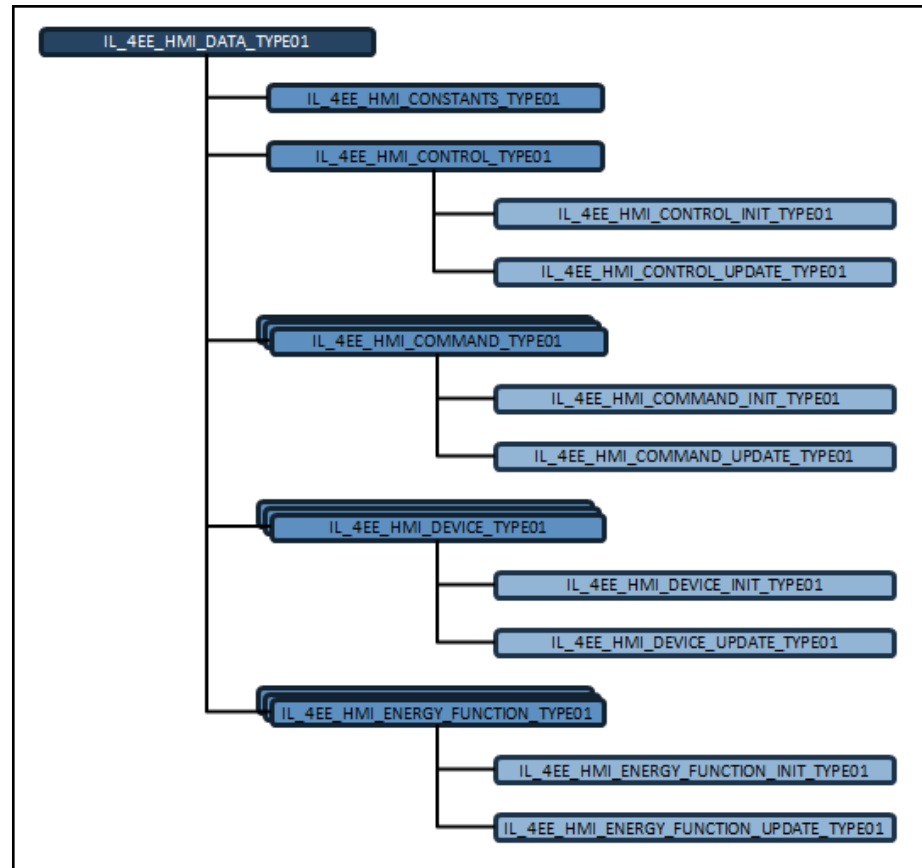


Fig. 9-88: Design of IL_4EE_HMI_DATA_TYPE01 structure

Name	Type	Description
Constants	IL_4EE_HMI_CONSTANTS_TYPE01	Array limits used in the structure see IL_4EE_HMI_CONSTANTS_TYPE01 , p. 271
Control	IL_4EE_HMI_CONTROL_TYPE01	Data for visualizing the EnergyControl block see IL_4EE_HMI_CONTROL_TYPE01 , p. 271
Command	ARRAY [1..NO_OF_COMMANDS_PER_HMI] OF IL_4EE_HMI_COMMAND_TYPE01	Data for visualizing the command blocks see Parameter list , p. 297, IL_4EE_HMI_COMMAND_TYPE01 , p. 273
Device	ARRAY [1..NO_OF_DEVICES_PER_HMI] OF IL_4EE_HMI_DEVICE_TYPE01	Data for visualizing the device blocks see Parameter list , p. 297, IL_4EE_HMI_DEVICE_TYPE01 , p. 274
EnergyFunction	ARRAY [1..NO_OF_ENERGY_FUNCTIONS_PER_HMI] OF IL_4EE_HMI_ENERGY_FUNCTION_TYPE01	Data for visualizing the EnergyFunction blocks see Parameter list , p. 297, IL_4EE_HMI_ENERGY_FUNCTION_TYPE01 , p. 275

Tab. 9-211: Min./max. and default values of the IL_4EE_HMI_TYPE01 structure

IL_4EE_HMI_CONSTANTS_TYPE01

Brief Description The substructure contains the array limits used in the structure [IL_4EE_HMI_DATA_TYPE01](#), p. 269,.

Name	Type	Min. value	Max. value	Default value	Effective	Description
NoOfDevices	USINT					Number of device blocks read by the HMIDataCollector block
NoOfCommands	USINT					Number of command blocks read by the HMIDataCollector block
NoOfEnergy-Functions	USINT					Number of EnergyFunction blocks read by the HMIDataCollector block
NoOfEMPerEnergyControl	USINT					Number of energy modes per EnergyControl block
NoOfEMPerDevice	USINT					Number of energy modes per device block

Tab. 9-212: Min./max. and default values of the IL_4EE_HMI_CONSTANTS_TYPE01 structure

IL_4EE_HMI_CONTROL_TYPE01

Brief Description The structure contains the data for visualizing the EnergyControl block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
CommandIdx	USINT					Shows the index of the command block which currently specifies the EnergyControl block command 0: No command block currently active 1-255: Each command block specifies the current command
Init	IL_4EE_HMI_CONTROL_INIT_TYPE01					Data updated while initializing the block see IL_4EE_HMI_CONTROL_INIT_TYPE01 , p. 272
Update	IL_4EE_HMI_CONTROL_UPDATE_TYPE01					Data updated while operating the block see IL_4EE_HMI_CONTROL_UPDATE_TYPE01 , p. 272

Tab. 9-213: Min./max. and default values of the IL_4EE_HMI_CONTROL_TYPE01 structure

IL_4EE_HMI_CONTROL_EM_TYPE01

The structure contains the data for configuring the energy modes.

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
EMIdent	USINT					ID of the energy mode
EMName	STRING(16)					Name of the energy mode
PowerConsumption	DINT					Average energy consumption in the energy mode

Tab. 9-214: Min./max. and default values of the IL_4EE_HMI_CONTROL_EM_TYPE01 structure

IL_4EE_HMI_CONTROL_INIT_TYPE01

Brief Description The structure contains data updated while initializing the block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
EM	ARRAY [0..NO_OF_EM_PER_ENERGY_CONTROL] OF IL_4EE_HMI_CONTROL_EM_TYPE01					Configuration of energy modes see Parameter list , p. 297, IL_4EE_HMI_CONTROL_EM_TYPE01 , p. 271

Tab. 9-215: Min./max. and default values of the IL_4EE_HMI_CONTROL_INIT_TYPE01 structure

IL_4EE_HMI_CONTROL_UPDATE_TYPE01

Brief Description The structure contains data updated while operating the block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
InOperation	BOOL					The EnergyControl block is in the InOperation state
ChangeEM	IL_4EE_CHANGE_EM					The energy mode should be changed see IL_4EE_CHANGE_EM , p. 269
PauseTime	UDINT					Currently specified dead time [s]
PowerLimit	UDINT					Currently specified power limit [Ws]
CmdTargetEM	USINT					ID of the target energy mode in the specification (only for Direct Mode)
TypeSel	IL_4EE_TYPE_SEL					Type of the current command 1: DirectMode 2: PauseTimeMode 3: PauseTimeWakeUpMode see IL_4EE_TYPE_SEL , p. 269

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
TargetEM	USINT					ID of the target energy mode
CurrentEM	USINT					ID of the current energy mode
RequEToEMOp	DINT					Energy consumption [Ws] needed to operational energy mode (dynamic)
RequEToEMx	DINT					Energy consumption [Ws] needed to target energy mode (dynamic)
TimeToEMOp	UINT					Remaining time [s] to operational energy mode (dynamic)
TimeToEMx	UINT					Remaining time [s] to target energy mode (dynamic)
CurrentPower-Consumption	DINT					Current energy consumption [W] (dynamic)
Error	BOOL					The block is in the error state
ErrorIdentAdditional1	DWORD					Error number for detailed diagnostics
ErrorIdentAdditional2	DWORD					Error number for detailed diagnostics

Tab. 9-216: Min./max. and default values of the IL_4EE_HMI_CONTROL_TYPE01 structure

IL_4EE_HMI_COMMAND_TYPE01

Brief Description The structure contains the data for visualizing the command blocks.

Name	Type	Min. value	Max. value	Default value	Effective	Description
DeviceIdx	USINT					Corresponding block: 0: Block registered with Energy-Control block 1-255: Block registered with respective device block
Init	IL_4EE_HMI_COMMAND_INIT_TYPE01					Data updated while initializing the block see IL_4EE_HMI_COMMAND_INIT_TYPE01 , p. 273
Update	IL_4EE_HMI_COMMAND_UPDATE_TYPE01					Data updated while operating the block see IL_4EE_HMI_COMMAND_UPDATE_TYPE01 , p. 274

Tab. 9-217: Min./max. and default values of the IL_4EE_HMI_COMMAND_DATA_TYPE01 structure

IL_4EE_HMI_COMMAND_INIT_TYPE01

Brief Description The structure contains data updated while initializing the block.

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
Visible	BOOL					Indicates whether or not the command should be shown in the EventState control
CommandName	STRING(16)					Name of the command block
DelayTime	UINT					Delay time [s] until command becomes active

Tab. 9-218: Min./max. and default values of the IL_4EE_HMI_COMMAND_INIT_TYPE01 structure

IL_4EE_HMI_COMMAND_UPDATE_TYPE01

Brief Description The structure contains data updated while operating the block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
InOperation	BOOL					Command block is operating
TargetEM	USINT					ID of the target energy mode
State	IL_4EE_CMD_STATE					Current status of the command See chapter "IL_4EE_CMD_STATE" on page 269
RemainingTime	UINT					Remaining time [s] until command becomes active

Tab. 9-219: Min./max. and default values of the IL_4EE_HMI_COMMAND_UPDATE_TYPE01 structure

IL_4EE_HMI_DEVICE_TYPE01

Brief Description The structure contains the data for visualizing the device blocks.

Name	Type	Min. value	Max. value	Default value	Effective	Description
Init	IL_4EE_HMI_DEVICE_INIT_TYPE01					Data updated while initializing the block see IL_4EE_HMI_DEVICE_INIT_TYPE01 , p. 274
Update	IL_4EE_HMI_DEVICE_UPDATE_TYPE01					Data updated while operating the block see IL_4EE_HMI_DEVICE_UPDATE_TYPE01 , p. 275

Tab. 9-220: Min./max. and default values of the IL_4EE_HMI_DEVICE_TYPE01 structure

IL_4EE_HMI_DEVICE_INIT_TYPE01

Brief Description The structure contains data updated while initializing the block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
DeviceName	STRING(16)					Name of the device
EM	IL_4EE_HMI_DE- VICE_EM_TYPE01					Configuration data for the device see IL_4EE_HMI_DE- VICE_EM_TYPE01 , p. 275

Tab. 9-221: Min./max. and default values of the IL_4EE_HMI_DEVICE_IN-IT_TYPE01 structure

IL_4EE_HMI_DEVICE_EM_TYPE01

Brief Description The structure contains the configuration data for the device

Name	Type	Min. value	Max. value	Default value	Effective	Description
EMIdent	USINT					ID of the device
EMName	STRING(16)					Name of the energy mode

Tab. 9-222: Min./max. and default values of the IL_4EE_HMI_DE-VICE_EM_TYPE01 structure

IL_4EE_HMI_DEVICE_UPDATE_TYPE01

Brief Description The structure contains data updated while operating the block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
ChangeEM	IL_4EE_CHANGE_EM					The energy mode for the device should be changed see IL_4EE_CHANGE_EM , p. 269
TargetEM	USINT					ID of the current target energy mode
CurrentEM	USINT					ID of the current energy mode

Tab. 9-223: Min./max. and default values of the IL_4EE_HMI_DEVICE_UPDATE_TYPE01 structure

IL_4EE_HMI_ENERGY_FUNCTION_TYPE01

Brief Description The structure contains substructures with the data for visualizing the Energy-Function blocks.

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective	Description
Init	IL_4EE_HMI_ENERGY_FUNCTION_INIT_TYPE01					Data updated while initializing the block see IL_4EE_HMI_ENERGY_FUNCTION_INIT_TYPE01 , p. 276
Update	IL_4EE_HMI_ENERGY_FUNCTION_UPDATE_TYPE01					Data updated while operating the block see IL_4EE_HMI_ENERGY_FUNCTION_UPDATE_TYPE01 , p. 276

Tab. 9-224: Min./max. and default values of the IL_4EE_ENERGY_FUNCTION_INIT_TYPE01 structure

IL_4EE_HMI_ENERGY_FUNCTION_INIT_TYPE01

Brief Description The structure contains data updated while initializing the block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
FunctionName	STRING(16)					Name of the energy saving function
DeviceIdx	USINT					Number of the corresponding device

Tab. 9-225: Min./max. and default values of the IL_4EE_ENERGY_FUNCTION_INIT_TYPE01 structure

IL_4EE_HMI_ENERGY_FUNCTION_UPDATE_TYPE01

Brief Description The structure contains data updated while operating the block.

Name	Type	Min. value	Max. value	Default value	Effective	Description
FunctionValue	INT					Current value of the energy saving function

Tab. 9-226: Min./max. and default values of the IL_4EE_HMI_ENERGY_FUNCTION_UPDATE_TYPE01 structure

9.4 PROFIenergy

9.4.1 Function blocks

IL_4EEProfiEnergyServiceType01

Brief description The IL_4EEProfiEnergyServiceType01 function block is used to execute the services specified in PROFIenergy. Use these services to switch commands and to read configuration, status and measuring data. Apart from the regular use via the inputs and outputs of the function block, the function block is also used by the IL_4EEMonProfiEnergyDeviceType01 and IL_4EEProfiEnergyDeviceType01 function blocks as communication function block. This is required as the PROFIenergy profile is not multi-client-compatible. This means that the function blocks for EnergyMonitoring and EnergyControl cannot simultaneously access the PROFIenergy device services. This is solved via a se-

RIL_4EEEnergyManagement.library

rialization of the service calls of the different clients in this function block. If the function block is used directly, only the inputs for the respective service have to be connected.

Target system	Library
MTX from 14VRS	RIL_4EEEnergyManagement
MLC from 14VRS	RIL_4EEEnergyManagement

Tab. 9-227: Assignment target system/library

Interface description

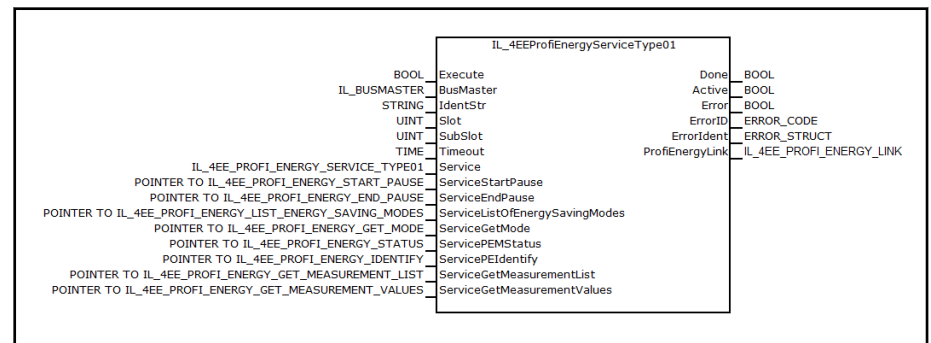


Fig. 9-89: IL_4EEProfEnergyServiceType01 function block

I/O type	Name	Data type	Description
VAR_INPUT	Execute	BOOL	Processing of the function block enabled (edge-controlled)
	BusMaster	IL_BUSMASTER	Instance of the field bus master in the order of the configuration tree
	IdentStr	STRING	Identification string of the field bus device
	Slot	UINT	Addressing of the module slot. A slot addresses a logical or a physical module of a Profinet device (e.g. an I/O module)
	SubSlot	UINT	Addressing of the submodule. A module can consist of several logical or physical submodules. At least the submodule with number 1 is present in each module (slot)
	Timeout	TIME	Timeout monitoring of the function block in ms T#0ms = Timeout monitoring not active
	Service	IL_4EE_PROFI_ENERGY_SERVICE_TYPE01	Selection of the PROFIenergy command that is to be executed
	ServiceStart-Pause	POINTER TO IL_4EE_PROFI_ENERGY_START_PAUSE	Input and output values for PROFIenergy service Start_Pause
	ServiceEnd-Pause	POINTER TO IL_4EE_PROFI_ENERGY_END_PAUSE	Output values for the PROFIenergy service End_Pause

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	ServiceListEnergySavingModes	POINTER TO IL_4EE_PROFI_ENERGY_END_PAUSE	Output values for the PROFIenergy service List_Energy_Saving_Modes
	ServiceGet-Mode	POINTER TO IL_4EE_PROFI_ENERGY_GET_MODE	Input and output values for PROFIenergy service Get_Mode
	ServicePEM-Status	POINTER TO IL_4EE_PROFI_ENERGY_STATUS	Output values for the PROFIenergy service PEM_Status
	ServicePEI-identify	POINTER TO IL_4EE_PROFI_ENERGY_IDENTIFY	Output values for the PROFIenergy service PE_Identify
	ServiceGet-Measurement-List	POINTER TO IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_LIST	Input and output values for PROFIenergy service Get_Measurement_List
	ServiceGet-Measurement-Values	POINTER TO IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_VALUES	Input and output values for PROFIenergy service Get_Measurement_Values
VAR_OUTPUT	Done	BOOL	Processing completed by the function block without error
	Active		Block is being processed
	Error	BOOL	Processing completed with error
	ErrorID	ERROR_CODE	Diagnostic description in case of error

RIL_4EEEnergyManagement.library

I/O type	Name	Data type	Description
	ErrorIdent	ERROR_STRUCTURE	Detailed diagnostic, see Troubleshooting, page 284
	ProfiEnergy-Link	IL_4EE_PROFI_ENERGY_LINK	Handle on the PROFIenergy services is required by the PROFIenergy device function blocks

Tab. 9-228: IL_4EEProfiEnergyServiceType01 function block

Name	Type	Min. value	Max. value	Default value	Effective
Execute	BOOL			FALSE	Pos. Execute edge
BusMaster	IL_BUSMASTER	IL_BUSMASTER_0	IL_BUSMASTER_4	IL_BUSMASTER_0	Pos. Execute edge
IdentStr	STRING				Pos. Execute edge
Slot	UINT	0	16#7FFF	0	Pos. Execute edge
SubSlot	UINT	0	16#8FFF	0	Pos. Execute edge
Timeout	TIME			T#0S	Pos. Execute edge
Service	IL_4EE_PROFI_ENERGY_SERVICE_TYPE01			PE_START_PAUSE	Pos. Execute edge
ServiceStartPause	POINTER TO IL_4EE_PROFI_ENERGY_START_PAUSE				Continuous
ServiceEndPause	POINTER TO IL_4EE_PROFI_ENERGY_END_PAUSE				Continuous
ServiceListEnergy-SavingModes	POINTER TO IL_4EE_PROFI_ENERGY_END_PAUSE				Continuous
ServiceGetMode	POINTER TO IL_4EE_PROFI_ENERGY_GET_MODE				Continuous
ServicePEMStatus	POINTER TO IL_4EE_PROFI_ENERGY_STATUS				Continuous

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Effective
ServicePEIdentify	POINTER TO IL_4EE_PROFI_ENERGY_IDENTIFY				Continuous
ServiceGetMeasurementList	POINTER TO IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_LIST				Continuous
ServiceGetMeasurementValues	POINTER TO IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_VALUES				Continuous

Tab. 9-229: Min./max. and default values of the IL_4EEProfiEnergyService-Type01 inputs

Functional description

The device is addressed via the Profinet device name in "IdentStr". It must be part of the configuration of the controller. The data set on the PROFINET device is addressed via the slot-subslot combination. Slot and subslot address the various logical or physical modules of a device.

The index is specified as 16#80A0 with the PROFInergy service access point (PESAP).

The PROFInergy service is to be selected via the Service input. It can be executed by starting the function block by a positive edge at Execute. Only the structure for the command to be executed has to be created via the pointer. If a command is not to be executed, the corresponding structure is not required.

Overview on the services:

The service **Start_Pause** is used to switch from the operating energy mode to another energy saving mode or to switch from one energy saving mode to another. Input variable is always the idle time used to calculate the best energy mode. Output variable is the ID number of the calculated energy mode.

The service **End_Pause** is used to switch from one energy mode to the operating energy mode or to cancel switching from the operating energy mode to an energy saving mode and back to the operating energy mode. No input parameter is required. The time required to switch to the operating energy mode is returned as output variable.

The **List_Energy_Saving_Modes** service is used to read out the ID numbers of the energy modes of the device. The number of ID numbers as well as an array with the ID numbers is the output variable.

The **Get_Mode** service is used to read out the configuration of the energy mode of the device. The input value is the ID number of the energy mode. Output values:

- The minimum idle time for the energy mode
- The switching time for the energy saving mode
- The switching time to the operating energy mode

- Minimum dwell time in the energy mode
- Minimum dwell time in the energy mode
- The average power consumption in the energy mode
- The energy required to change to the operating energy mode
- The energy required to change to the energy saving mode

These values can be static as well as dynamic. This is also displayed via an output value.

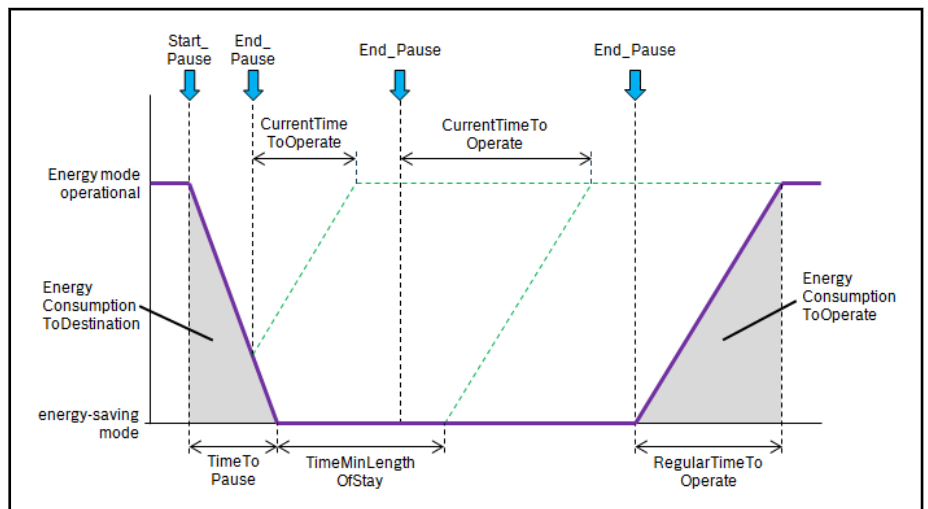


Fig. 9-90: Representation of configuration data of an energy mode

The **PEM_Status** service is used to read out the current status of device switching or of the current energy mode. Output variables:

- ID number of the original energy mode
- ID number of the target energy mode
- Time required to switch to operating energy mode
- Time required to switch to target energy mode
- Average power in the current energy mode
- Energy required to switch to target energy mode
- Energy required to switch to operating energy mode

The time and energy data can be static as well as dynamic.

The **PE_Identify** service is used to read out the services supported by the device. The output variables are the number of supported services as well as an array with ID numbers of the output variables.

Identification number	Contained services
16#01	Start_Pause
16#02	End_Pause
16#03	List_Energy_Saving_Modes
	Get_Mode
16#04	PEM_Status
16#05	PE_Identify

RIL_4EEEnergyManagement.library

16#10	Get_Measurement_List
	Get_Measurement_Values

Tab. 9-230: ID numbers of the ProfiEnergyLink services

The **Get_Measurement_List** service is used to read out the ID numbers as well as the measured variables supported by the device and their accuracy classes. The output variables are the number of measured variables as well as an array of structures containing the ID numbers and accuracy classes of the output variables.

The measured variables are divided into the following groups:

- Current values (ID number 1 - 100)
- Mean values (150 - 199 and from 230)
- Energy values (200 - 229)

The most important values are:

Identi- fica- tion num- ber	Description	Unit	Data type	Message dura- tion Typical values
1	Voltage L1 - N	V	REAL	-
2	Voltage L2 - N	V	REAL	-
3	Voltage L3 - N	V	REAL	-
4	Voltage L1 - L2	V	REAL	-
5	Voltage L2 - L3	V	REAL	-
6	Voltage L3 - L1	V	REAL	-
7	Current L1	A	REAL	-
8	Current L2	A	REAL	-
9	Current L3	A	REAL	-
34	Effective power	W	REAL	-
150	Voltage L1 - N	V	REAL	3 s
151	Voltage L2 - N	V	REAL	3 s
152	Voltage L3 - N	V	REAL	3 s
153	Voltage L1 - L2	V	REAL	3 s
154	Voltage L2 - L3	V	REAL	3 s
155	Voltage L3 - L1	V	REAL	3 s
156	Current L1	A	REAL	600s
157	Current L2	A	REAL	600s
158	Current L3	A	REAL	600s
163	Effective power	W	REAL	900s
200	Active energy input	Wh	REAL	-
201	Active energy output	Wh	REAL	-

RIL_4EEEnergyManagement.library

205	Total sum active energy	Wh	REAL	-
210	Active energy input	Wh	LREAL	-
211	Active energy output	Wh	LREAL	-
215	Total sum active energy	Wh	LREAL	-
220	Active energy input	Wh	DINT	-
221	Active energy output	Wh	DINT	-
225	Total sum active energy	Wh	DINT	-

Tab. 9-231: Overview on the different measured values

The accuracy class is specified as AccuracyDomain (domain of definition) and AccuracyClass.

Accuracy Class	Accuracy Domain 1 or 2	Accuracy Domain 3	Accuracy Domain 4
0	Reserved	Reserved	Reserved
1	0.01%	0.02	0.5
2	0.02%	0.05	1.0
3	0.05%	0.1	1.5
4	0.1%	0.2	2.0
5	0.2%	0.5	2.5
6	0.5%	1	3
7	1%	1.5	undefined
8	1.5%	2	
9	2%	2.5	
10	2.5%	3	
11	3%	5	
12	5%	10	
13	10%	20	
14	20%	undefined	
15	>20%		
>15	undefined		

Tab. 9-232: Accuracy classes of the measured values

For AccuracyDomain 1 and 2, the measuring range is additionally specified for calculation.

The **Get_Measurement_Values** service is used to read out up to 12 measured variables. Input variable is an array with the ID numbers to be read out. Output variables are the status of the measured values, the values of the measured variables, converted to the data type LREAL and, if available, the time stamp of the measured value.

Use with other function blocks

The function block is used in connection with the PROFIenergy device function blocks (e.g. IL_4EEMonProfiEnergyDeviceType01). This is required as the mechanisms used for PROFIenergy are not multi-client-compatible. This means, if several instances of IL_4EEProfiEnergyServiceType01 access the

RIL_4EEEnergyManagement.library

PROFIenergy services of a device, they are interfering with each other. Thus, the function block provides a handle. Via this handle, an instance of the IL_4EEMonProfiEnergyDeviceType01 function block can call the PROFIenergy services. The IL_4EEMonProfiEnergyServiceType01 implements a serialization of the accesses to the PROFIenergy services by the different clients. Apart from the user interface, up to three other instances can use the services. This also facilitates the use of EnergyMonitoring and EnergyControl.

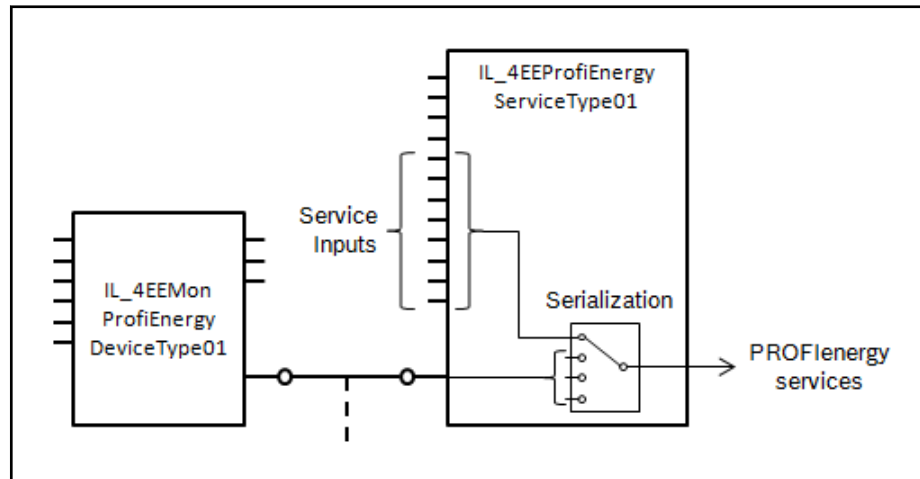


Fig. 9-91: Serialization of services

Troubleshooting

The function block outputs the following error messages of F_RELATED_TABLE. Additionally, error messages of the IL_PNIOServiceRecord function block can occur.

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR	16#2A20	16#0000	Cause: The "Service" input has an invalid value Solution: Set input to a valid enumeration value
INPUT_INVALID_ERROR	16#2A20	16#0001	Cause: The "ServiceStartPause" input is a zero pointer and the service "Start_Pause" was selected Solution: Create a pointer to a structure of type IL_4EE_PROFI_ENERGY_START_PAUSE
INPUT_INVALID_ERROR	16#2A20	16#0002	Cause: The "ServiceEndPause" input is a zero pointer and the service "End_Pause" was selected Solution: Create a pointer to a structure of type IL_4EE_PROFI_ENERGY_END_PAUSE
INPUT_INVALID_ERROR	16#2A20	16#0003	Cause: The "ServiceListEnergySavingModes" input is a zero pointer and the service "List_Energy_Saving_Modes" was selected Solution: Create a pointer to a structure of type IL_4EE_PROFI_ENERGY_LIST_ENERGY_SAVING_MODES
INPUT_INVALID_ERROR	16#2A20	16#0004	Cause: The ServiceGetMode input is a zero pointer and the service Get_Mode was selected Solution: Create a pointer to a structure of type IL_4EE_PROFI_ENERGY_GET_MODE

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
INPUT_INVALID_ERROR	16#2A20	16#0005	Cause: The ServicePEMStatus input is a zero pointer and the service PEM_Status was selected Solution: Create a pointer to a structure of type IL_4EE_PROFI_ENERGY_STATUS
INPUT_INVALID_ERROR	16#2A20	16#0006	Cause: The ServicePEIdentify input is a zero pointer and the service PE_Identify was selected Solution: Create a pointer to a structure of type IL_4EE_PROFI_ENERGY_IDENTIFY
INPUT_INVALID_ERROR	16#2A20	16#0007	Cause: The ServiceGetMeasurementList input is a zero pointer and the service Get_Measurement_List was selected Solution: Create a pointer to a structure of type IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_LIST
INPUT_INVALID_ERROR	16#2A20	16#0008	Cause: The ServiceGetMeasurementList^ MeasIdentList is a zero pointer and the service Get_Measurement_List has been selected Solution: Create a pointer to an array of type IL_4EE_PROFI_ENERGY_MEASUREMENT_ID
INPUT_INVALID_ERROR	16#2A20	16#0009	Cause: The size of the array specified via ServiceGetMeasurementList^. SizeOfMeasIdentList is not sufficient Solution: The minimum length can be read out via MethGetMeasurementList^.MeasIdentListCount
INPUT_INVALID_ERROR	16#2A20	16#000A	Cause: The ServiceGetMeasurementValues input is a zero pointer and the service Get_Measurement_Values was selected Solution: Create a pointer to a structure of type IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_VALUES
RESOURCE_ERROR	16#2A21	16#0000	Cause: The feedback data is incomplete (PROFIenergy status: "Data incomplete") Solution: Cannot be influenced
RESOURCE_ERROR	16#2A21	16#0001	Cause: The service feedback is not part of the query. This can happen when trying to access a device with several instances Solution: Use only one instance of the function block per device
RESOURCE_ERROR	16#2A21	16#0101	Cause: The selected service is not supported by the device Solution: Verify the supported services using the service PE_Identify
RESOURCE_ERROR	16#2A21	16#0102	Cause: Invalid service reference no. Solution: Error in the function block or in the device

RIL_4EEEnergyManagement.library

ErrorID	Additional1	Additional2	Description
RESOURCE_ERROR	16#2A21	16#0103	Cause: The selected service is not supported by the device Solution: Error in the function block or in the device
RESOURCE_ERROR	16#2A21	16#0104	Cause: The service has not been selected correctly Solution: Error in the function block or in the device
RESOURCE_ERROR	16#2A21	16#0105	Cause: The service has not been selected correctly Solution: Error in the function block or in the device
RESOURCE_ERROR	16#2A21	16#0106	Cause: No energy saving mode is supported Solution: EnergyControl functions cannot be used
RESOURCE_ERROR	16#2A21	16#0107	Cause: The length of the response is too long Solution: Error in the function block or in the device
RESOURCE_ERROR	16#2A21	16#0108	Cause: The header is incorrect Solution: Error in the function block or in the device
RESOURCE_ERROR	16#2A21	16#0150	Cause: No compatible energy saving mode has been found in the Start_Pause serve Solution: The device remains in operating energy mode for the specified dead time and only switches off during longer idle times
RESOURCE_ERROR	16#2A21	16#0151	Cause: Time is not supported Solution: Problem in the device
RESOURCE_ERROR	16#2A21	16#0152	Cause: An energy mode that is not available has been selected in the Get_Mode service Solution: Verify energy modes using the List_Energy_Saving_Modes service
RESOURCE_ERROR	16#2A21	16#0153	Cause: When using the Start_Pause service, switching to energy saving mode is not possible as the device is still operated Solution: Stop device
RESOURCE_ERROR	16#2A21	16#0154	Cause: The service or the function can currently not be used (maybe device is in setup) Solution: Retry after a waiting time
OTHER_ERROR	16#2A22	16#0000	Cause: The constant NO_OF_EM_PER_DEVICE is too small so that not all energy modes can be read out Solution: Adjust the value of the constant in Parameter list, page 297 . The number of energy modes can be read out via the Count element of the structure
OTHER_ERROR	16#2A22	16#0001	Cause: More services are available in the device than can be read out using this function block (max. 20)
STATE_MACHINE_ERROR	16#0006	16#FFFF	Invalid function block state

Tab. 9-233: Error codes function block IL_4EEProfEnergyServiceType01

9.4.2 Data types

IL_4EE_PROFI_ENERGY_START_PAUSE

Brief description The structure IL_4EE_PROFI_ENERGY_START_PAUSE contains the input and output data for the PROFIenergy service Start_Pause to switch the load to an energy-saving state.

Name	Type	Description
PauseTime	TIME	Expected idle time to calculate the energy mode (Block input)
TargetEM	USINT	The calculated target energy mode the load is switched to (Block output)

Tab. 9-234: Design of the IL_4EE_PROFI_ENERGY_START_PAUSE structure

IL_4EE_PROFI_ENERGY_END_PAUSE

Brief description The structure IL_4EE_PROFI_ENERGY_START_PAUSE contains the output data for the PROFIenergy service End_Pause to switch the load to the operating energy mode.

Name	Type	Description
CurrentTime-ToOperate	TIME	Expected duration for switching to operating energy mode (Block output)

Tab. 9-235: Design of the IL_4EE_PROFI_ENERGY_END_PAUSE structure

IL_4EE_PROFI_ENERGY_LIST_ENERGY_SAVING_MODES

Brief description The structure IL_4EE_PROFI_ENERGY_LIST_ENERGY_SAVING_MODES contains the output data for the PROFIenergy service List_Energy_Saving_Modes to read out the ID numbers of energy modes supported by the load. The constant NO_OF_EM_PER_DEVICE can be changed via the parameter list.

Name	Type	Description
Count	USINT	Number of energy modes supporting the consumer (Block output)
EMIdents	ARRAY [1..NO_OF_EM_PER_DEVICE] OF USINT	ID numbers of energy modes supporting the consumer (Block output)

Tab. 9-236: Design of the IL_4EE_PROFI_ENERGY_LIST_ENERGY_SAVING_MODES structure

IL_4EE_PROFI_ENERGY_GET_MODE

Brief description The structure IL_4EE_PROFI_ENERGY_GET_MODE contains the input and output data for the PROFIenergy service Get_Mode to read out the configuration of an energy mode.

RIL_4EEEnergyManagement.library

Name	Type	Description
EMIdent	USINT	ID number of the energy mode (Input and output of the block)
ModeAttributes	USINT	Bit-coded attribute of the energy mode Bit 0: 0 => Static time and energy data; 1 => Dynamic time and energy data (Block output)
TimeMinPause	TIME	Minimum idle time for this energy mode, sum of TimeToPause, RegularTimeToOperate and TimeMinLengthOfStay (Block output)
TimeToPause	TIME	Minimum idle time for this energy mode, sum of TimeToPause, RegularTimeToOperate and TimeMinLengthOfStay (Block output)
RegularTimeToOperate	TIME	Regular time to change back to the operating energy mode (Block output)
TimeMinLenghtOfStay	TIME	Minimum dwell time in the energy mode (Block output)
TimeMaxLengthOfStay	TIME	Maximum dwell time in the energy mode (Block output)
ModePowerConsumption	REAL	Power consumption in energy mode [kW] (Block output)
EnergyConsumptionToPause	REAL	Energy consumption during the change to this energy mode (Block output)
EnergyConsumptionToOperate	REAL	Energy consumption during the change to the operating energy mode (Block output)

Tab. 9-237: Design of the IL_4EE_PROFI_ENERGY_GET_MODE structure

IL_4EE_PROFI_ENERGY_STATUS

Brief description The structure IL_4EE_PROFI_ENERGY_STATUS contains the output data for PROFInergy service PEM_Status to read out the current state of the load.

Name	Type	Description
EMIdentSource	USINT	ID number of the energy mode (Block output)
EMIdentDestination	USINT	Bit-coded attribute of the energy mode Bit 0: 0 => Static time and energy data; 1 => Dynamic time and energy data (Block output)

RIL_4EEEnergyManagement.library

Name	Type	Description
CurrentTime-ToOperate	TIME	Minimum idle time for this energy mode, sum of Time-ToPause, RegularTimeToOperate and TimeMin-LengthOfStay (Block output)
CurrentTimeTo-Destination	TIME	Time needed to change to energy mode (Block output)
ModePowerCon-sumption	REAL	Regular time to change back to the operating energy mode (Block output)
EnergyCon-sumptionToDes-tination	REAL	Minimum dwell time in the energy mode (Block output)
EnergyCon-sumptionToO-perate	REAL	Maximum dwell time in the energy mode (Block output)

Tab. 9-238: Design of the IL_4EE_PROFI_ENERGY_STATUS structure

IL_4EE_PROFI_ENERGY_IDENTIFY

Brief description The structure IL_4EE_PROFI_ENERGY_IDENTIFY contains the output data for the PROFInergy service PE_Identify to read out the supported PROFInergy commands of the device.

Name	Type	Description
Count	USINT	Number of supported PROFInergy commands (Block output)
PEServices	ARRAY [1..NO_OF_SERVICES] OF USINT	ID numbers of the supported PROFInergy command (NO_OF_SERVICES = 20) (Block output)

Tab. 9-239: Design of the IL_4EE_PROFI_ENERGY_IDENTIFY structure

IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_LIST

Brief description The structure IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_LIST contains the output data for the PROFInergy service Get_Measurement_List to read out the supported measured variables of the device.

Name	Type	Description
SizeOfMeas-identList	UINT	Size of the array whose first field is transmitted with the pointer MeasIdentList (Block input)
MeasIdentList	POINTER TO IL_4EE_PROFI_ENERGY_MEASUREMENT_ID	First array field to input the ID numbers of the supported measured values (Block output)
MeasIdentList	UINT	Number of read ID numbers (Block output)

Tab. 9-240: Design of the IL_4EE_PROFI_ENERGY_END_PAUSE structure

RIL_4EEEnergyManagement.library

IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_VALUES

Brief description The structure IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_VALUES contains the output data for the PROFIenergy service Get_Measurement_Value to read out the values of the supported measured variables of the device. The maximum number is set to 12 via the constant IL_4EE_PROFI_ENERGY_MEAS_VALUE.

Name	Type	Description
Ident	ARRAY [1..NO_OF_MEAS_IDS] OF UINT	ID number of the read measured values (Block input)
Values	ARRAY [1..NO_OF_MEAS_IDS] OF IL_4EE_PROFI_ENERGY_MEAS_VALUE	Array of the read measured values (Block output)

Tab. 9-241: Design of the IL_4EE_PROFI_ENERGY_GET_MEASUREMENT_VALUES structure

IL_4EE_PROFI_ENERGY_MEAS_VALUE

Brief description The structure IL_4EE_PROFI_ENERGY_MEAS_VALUE contains the status of the measured value, the measured value and, if available, the time stamp, divided into the time of the day and the date.

Name	Type	Description
Status	USINT	Status of measured value; 1= valid; 2= not available; 3= currently not available
Value	LREAL	Measured value as LREAL
Timestamp1	TIME_OF_DAY	Time stamp of the measured value, time
Timestamp2	DATE	Time stamp of the measured value, date

Tab. 9-242: Design of the IL_4EE_PROFI_ENERGY_MEAS_VALUE structure

9.5 Data types for the MES/control system interface

9.5.1 MachineEnergyObjectType

The MachineEnergyObjectType structure provides the interface to the control system/MES. It is divided into the MachineEnergyCtrlType (EnergyControl) substructures and MachineEnergyMonType (EnergyMonitoring).

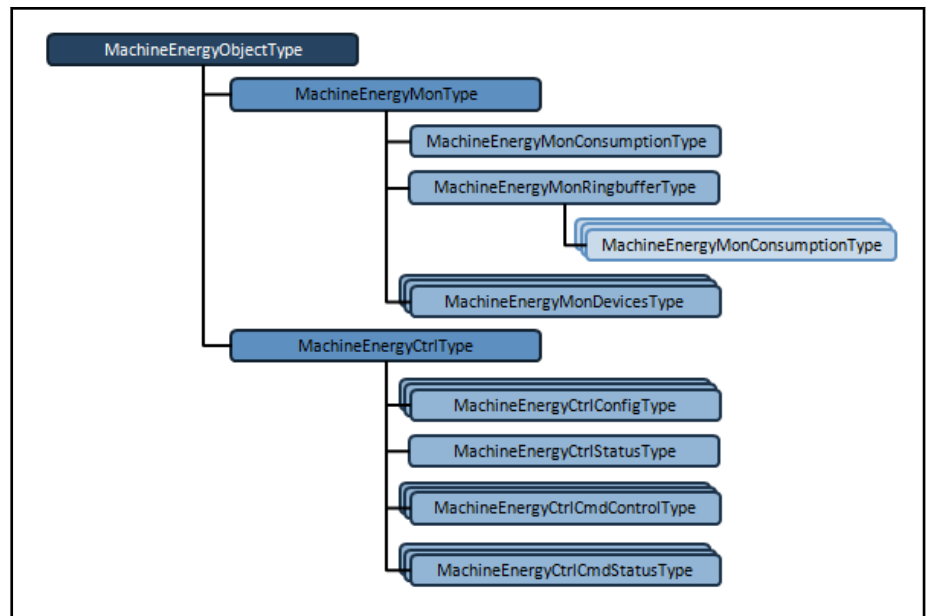


Fig. 9-92: Structure of the MachineEnergyObjectType data type

Brief description The MachineEnergyMon and MachineEnergyCtrl substructures are applied to the [IL_4EEMonScadaDataCollectorType01](#), page 190 and [IL_4EEScadaDataCollectorType01](#) function blocks, page 259. The ApplicationProfile element is generated by rounding the ApplicationProfile output values of both function blocks.

Name	Type	Min. value	Max. value	Default value	Description
Version	UINT			1	Interface version for evaluation of the control system/MES
ApplicationProfile	UDINT			0	Displays which substructures are supported (bit-coded) for evaluation by the control system/MES
MachineEnergyMon	MachineEnergyMonType				EnergyMonitorings data MachineEnergyMonType , page 291,
MachineEnergyCtrl	MachineEnergyCtrlType				EnergyControl data MachineEnergyCtrlType , page 293,

Tab. 9-243: Min./max. and default values of the MachineEnergyObjectType structure

9.5.2 MachineEnergyMonType

Brief description The MachineEnergyMonType structure is composed of the substructure of the machine, a ring buffer with several energy measurements as well as the energy consumption of the individual consumers. The structure is applied to the [IL_4EEMonScadaDataCollectorType01](#) function block, page 190.

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Description
MachineEnergyMonConsumption	MachineEnergyMonConsumptionType				Contains the current energy counter value MachineEnergyMonConsumptionType, page 292,
MachineEnergyMonRingbuffer	MachineEnergyMonRingbufferType				Contains several energy measurements MachineEnergyMonRingbufferType, page 293,
MachineEnergyMonDevices	ARRAY [1..NO_OF_MON_DEVICES_PER_ENERGY_MONITOR] OF MachineEnergyMonDevicesType				Contains the energy counter of the individual consumers Parameter list, page 297, MachineEnergyMonDevicesType, page 293,

Tab. 9-244: Min./max. and default values of the MachineEnergyMonType structure

9.5.3 MachineEnergyMonConsumptionType

Brief description The MachineEnergyMonConsumptionType structure contains the current energy counter value, the maximum value of the performance as well as data to assign the measured values to manufactured parts. The structure is part of the [MachineEnergyMonType structure, page 291](#) to display the current counter value as well the [MachineEnergyMonRingbufferType structure, page 293](#) to display the results of energy measurements. The structure is written by the [IL_4EEMonScadaDataCollectorType01 function block, page 190.](#)

Name	Type	Min. value	Max. value	Default value	Description
ConsumedEnergyOdometer	DINT			0	Current energy counter status of the machine [Ws] Ring buffer: Measured energy value [Ws] for the measurement
PowerPeakValue	REAL			0	Maximum power peak [W] as "Peak Hold with return movement" Ring buffer: Maximum power during the measurement
PartID	STRING(40)			"	ID of the currently manufactured part ring buffer: ID of the part manufactured during the measurement
PartDuration	TIME			T#0S	Duration to manufacture the last part Ring buffer: Duration for manufacturing during the measurement
PartCount	UINT			0	Counter for the manufactured products Ring buffer: Number of manufactured parts during the measurement

Tab. 9-245: Min./max. and default values of the MachineEnergyMonConsumptionType structure

9.5.4 MachineEnergyMonRingbufferType

Brief description The MachineEnergyMonRingbufferType structure contains the results of several time- or part-related energy measurements. The structure is written by the [IL_4EEMonScadaDataCollectorType01 function block, page 190](#).

The ring buffer is contained in the Consumption array. Via the BufferStart element, the index of the last completed measurement in the ring buffer is displayed. This index is filled by the function block with the increasing index. This means that the second to last measurement is stored in the index "BufferStart minus one" (exception: BufferStart has the value 1. The index then corresponds to the second to last measurement of the "NO_OF_MEAS_PER_SCADA") constant. The NoOfEntries element indicates how many measurements were stored in the ring buffer. This is needed after the function block start until not all indexes of the ring buffer are used.

Name	Type	Min. value	Max. value	Default value	Description
BufferMode	IL_4EE_RING_BUFFER_MODE			TIME_RELATED	Indicates if the energy measurements are time or part-related IL_4EE_RINGBUFFER_MODE, page 296 ,
SampleTime	TIME			T#0s	Time interval for time-related energy measurements
BufferStart	INT			0	Index of the ring buffer start (cursor)
NoOfEntries	INT			0	Number of measurements entered in the ring buffer
Consumption	ARRAY [1..NO_OF_RINGBUFFER_ENTRIES_PER_SCADA] OF MachineEnergyMonConsumptionType				Data of energy measurements Parameter list, page 297 , MachineEnergyMonConsumptionType, page 292 ,

Tab. 9-246: Min./max. and default values of the MachineEnergyMonRingbuffer-Type structure

9.5.5 MachineEnergyMonDevicesType

Brief description The MachineEnergyMonDeviceType structure contains the names and the energy counter values of the consumers. The structure is written by the [IL_4EEMonScadaDataCollectorType01 function block, page 190](#).

Name	Type	Min. value	Max. value	Default value	Description
DeviceName	STRING(16)				Name of consumer
ConsumedEnergyOdometer	DINT				Energy counter value of the consumer [Ws]

Tab. 9-247: Min./max. and default values of the MachineEnergyMonDevicesType structure

9.5.6 MachineEnergyCtrlType

Brief description The MachineEnergyCtrlType structure is composed of the configuration, the status and the EnergyControl. The structure is applied to the [IL_4EEScadaDataCollectorType01 function block, page 259](#).

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Description
MachineEnergyCtrlConfig	ARRAY [1..NO_OF_EM_PER_ENERGY_CONTROL + 1] OF MachineEnergyCtrlConfigType				Contains the configuration data of the energy modes of the machine Parameter list, page 297, MachineEnergyCtrlConfigType, page 294,
MachineEnergyCtrlStatus	MachineEnergyCtrlStatusType				Contains the current machine status MachineEnergyCtrlStatusType, page 295,
MachineEnergyCtrlCmdControl	ARRAY [1..NO_OF_COMMANDS_PER_SCADA] OF MachineEnergyCtrlCmdControlType				Contains the specifications of the control system/MES and is also written by the control system/MES Parameter list, page 297, MachineEnergyCtrlCmdControlType, page 295,
MachineEnergyCtrlCmdStatus	ARRAY [1..NO_OF_COMMANDS_PER_SCADA] OF MachineEnergyCtrlCmdStatusType				Contains the status of the control system/MES specifications Parameter list, page 297, MachineEnergyCtrlCmdStatusType, page 296,

Tab. 9-248: Min./max. and default values of the MachineEnergyCtrlType structure

9.5.7 MachineEnergyCtrlConfigType

Brief description The MachineEnergyCtrlConfigType structure contains the configuration of the energy modes for the EnergyControl. The structure is written by the [IL_4EES-cadaDataCollectorType01 function block, page 259.](#)

Name	Type	Min. value	Max. value	Default value	Description
EMIdent	USINT				ID number of the energy mode
EMValid	BOOL				TRUE: The energy mode has been configured FALSE: The energy mode was not taken into consideration
EMName	STRING(16)				Name of the energy mode
TimeToEMx	TIME				Time required for a change from the operating energy mode
TimeToEMOp	TIME				Time required for a change to the operating energy mode
TimeMinStay	TIME				Minimum dwell time in the energy mode
TimeMaxStay	TIME				Maximum dwell time in the energy mode

Name	Type	Min. value	Max. value	Default value	Description
PowerConsumption	REAL				Average energy consumption in the energy mode
RequEToEMx	REAL				Energy required for a change from the operating energy mode
RequEToEMOp	REAL				Energy required for a change to the operating energy mode

Tab. 9-249: Min./max. and default values of the MachineEnergyCtrlConfigType structure

9.5.8 MachineEnergyCtrlStatusType

Brief description The MachineEnergyCtrlStatusType structure contains the current status of the EnergyControl. The structure is written by the [IL_4EEscadaDataCollectorType01 function block, page 259](#).

Name	Type	Min. value	Max. value	Default value	Description
CurrentEM	USINT			0	Current energy mode
EMControlState	IL_4EE_ENERGY_CONTROL_STATE			EC_IDLE	State of the EnergyControl function block IL_4EE_ENERGY_CONTROL_STATE, page 296 ,
TypeSel	IL_4EE_TYPE_SEL			DIRECT_MODE	Type of the currently valid specification IL_4EE_TYPE_SEL, page 269 ,
PauseTime	TIME			T#0S	Dead time of the currently valid specification
TargetEM	USINT			0	Target energy mode of the currently valid specification
TimeToEMOp	TIME			T#0S	Time to switch to operating energy mode
TimeToEMx	TIME			T#0S	Time to switch to the current target energy mode

Tab. 9-250: Min./max. and default values of the MachineEnergyCtrlStatusType structure

9.5.9 MachineEnergyCtrlCmdControlType

Brief description Specifications for dead times can be specified from the control system/MES to the machine via the MachineEnergyCtrlCmdControlType structure. The structure is read by the [IL_4EEscadaDataCollectorType01 function block, page 259](#).

Name	Type	Min. value	Max. value	Default value	Description
ChangeEM	IL_4EE_CHANGE_EM			NO_CHANGE	Variable to trigger the specification IL_4EE_CHANGE_EM, page 269 ,
TypeSel	IL_4EE_TYPE_SEL			DIRECT_MODE	Variable to select the specification type IL_4EE_TYPE_SEL, page 269 ,

RIL_4EEEnergyManagement.library

Name	Type	Min. value	Max. value	Default value	Description
CmdTargetEM	USINT			0	Target energy mode for DirectMode
PauseTime	TIME			T#0s	Dead time for PauseTime mode and PauseTimeWakeUpMode

Tab. 9-251: Min./max. and default values of the MachineEnergyCtrlCmdControl-Type structure

9.5.10 MachineEnergyCtrlCmdStatusType

Brief description The status of the dead time specifications is indicated to the machine by the control system/MES via the MachineEnergyCtrlCmdStatusType structure. The structure is written by the [IL_4EEscadaDataCollectorType01 function block](#), page 259.

Name	Type	Min. value	Max. value	Default value	Description
CmdState	IL_4EE_CMD_STATE				Status of the corresponding specification IL_4EE_CMD_STATE , page 269,
ChangeEMack	IL_4EE_CHANGE_EM				Handshake for the MachineEnergyCtrlCmdControl-Type variable. ChangeEMack indicates the current variable value on the control IL_4EE_CHANGE_EM , page 269,
CmdDiagnosis	UDINT				Error Code, if specification cannot be edited

Tab. 9-252: Min./max. and default values of the MachineEnergyCtrlCmdStatus-Type structure

9.5.11 IL_4EE_RINGBUFFER_MODE

Brief description This enumeration is used to differentiate between different energy measurement modes such as time-related and part-related measurements.

Element	Value	Description
TIME_RELATED	0	Time-related energy measurements
PART_RELATED	1	Part-related energy measurements

Tab. 9-253: Elements of the IL_4EE_RINGBUFFER_MODE enumeration type

9.5.12 IL_4EE_ENERGY_CONTROL_STATE

Brief description This enumeration is used to indicate the current state of the EnergyControl function block.

Element	Value	Description
EC_IDLE	0	No specification active
EC_ACTIVE	1	Machine is switching between two energy modes
EC_INEM	2	Machine is in target energy mode, specification continues to be active
EC_WAKE_UP_MODE	3	Machine is in WakeUp mode, switching to the operating energy mode is active

Element	Value	Description
EC_TIME_MIN_STAY	4	Machine is in minimum dwell time of the current energy mode
EC_TIME_MAX_STAY	5	The maximum dwell time of the energy mode has expired. The machine switches back to operating energy mode

Tab. 9-254: Elements of the IL_4EE_ENERGY_CONTROL_STATE enumeration type

9.6 Parameter list

Constant	Type	Default value	Max. value	Description
NO_OF_EM_PER_ENERGY_CONTROL	USINT	8		Number of configurable energy modes for an EnergyControl block
NO_OF_EM_PER_DEVICE	USINT	4		Number of configurable energy modes for a device block
NO_OF_DEVICES_PER_ENERGY_CONTROL	USINT	32		Number of device and EnergyControl blocks that can be registered with an EnergyControl block
NO_OF_COMMANDS_PER_ENERGY_CONTROL	USINT	16		Number of command blocks that can be registered with an EnergyControl block
NO_OF_COMMANDS_PER_DEVICE	USINT	4		Number of command blocks that can be registered with a device block
NO_OF_DEVICES_PER_HMI	USINT	32		Number of device blocks visualized by an HMIDataCollector block
NO_OF_COMMANDS_PER_HMI	USINT	32		Number of command blocks visualized by an HMIDataCollector block
NO_OF_ENERGY_FUNCTIONS_PER_HMI	USINT	0		Number of EnergyFunction blocks visualized by an HMIDataCollector block
NO_OF_COMMANDS_PER_SCADA	USINT	2		Number of command blocks contained in a ScadaDataCollector function block
NO_OF_MON_DEVICES_PER_ENERGY_MONITOR	USINT	84		Number of MonitoringDevice blocks that can be registered with an EnergyMonitor block
NO_OF_RINGBUFFER_ENTRIES_PER_SCADA	USINT	4		Number of measurements that can be saved in the ring buffer of the interface for the MES/control system
IL_4EE_NO_OF_DRIVES	INT	64	64	Number of drives that can be visualized by Monitoring
IL_4EE_NO_OF_UNITS	INT	20	20	Number of units that can be visualized by Monitoring

Tab. 9-255: Parameter list constants

10 Service and support

Our worldwide service network provides an optimized and efficient support. Our experts offer you advice and assistance should you have any queries. You can contact us **24/7**.

Service Germany Our technology-oriented Competence Center in Lohr, Germany, is responsible for all your service-related queries for electric drive and controls.

Contact the **Service Hotline** and **Service Helpdesk** under:

Phone: **+49 9352 40 5060**
Fax: **+49 9352 18 4941**
E-mail: service.svc@boschrexroth.de
Internet: <http://www.boschrexroth.com>

Additional information on service, repair (e.g. delivery addresses) and training can be found on our internet sites.

Service worldwide Outside Germany, please contact your local service office first. For hotline numbers, refer to the sales office addresses on the internet.

Preparing information To be able to help you more quickly and efficiently, please have the following information ready:

- Detailed description of malfunction and circumstances
- Type plate specifications of the affected products, in particular type codes and serial numbers
- Your contact data (phone and fax number as well as your e-mail address)

Index

Symbols

- mSetCylinderPosition	
IL_4EEMonPneumaticDeviceType01.....	171
- mSetPressure	
IL_4EEMonPneumaticDeviceType01.....	171

A

Abbreviations.....	17
About this documentation.....	5
Validity of the documentation.....	5
Advanced functionality	
EnergyMonitor.....	79
PowerMonitor.....	81
ANSI Z535.6-2006.....	6
Application phases.....	5

C

Commissioning example: Handling and press....	31
Overview.....	31
Commissioning the EnergyMonitor.....	79
Requirements.....	79
Commissioning the PowerMonitor.....	82
Requirements.....	82
Commissioning visualization	
Use.....	52
Complaints.....	18
Component information.....	86
Components.....	79, 81, 85
Configuring the BasicDevice function blocks.....	65
Configuring the Command function blocks.....	61
Configuring the EnergyControl.....	56
Overview.....	56
Configuring the EnergyControl function blocks.....	56
Configuring the EnergyManagement function blocks.....	35
Overview.....	35
Configuring the EnergyMonitor block.....	50
Configuring the EnergyMonitoring.....	50
Overview.....	50
Configuring the HMIDataCollector block.....	50
Configuring the MonitoringDevice blocks.....	50
Connecting axes for MLC.....	69
Connecting conveyor belts and lighting.....	69
Connecting the axes for MTX.....	70
Connecting the consumers to EnergyManagement.....	68
Overview.....	68
Connecting the consumers to EnergyMonitoring.....	51
Coupling.....	29
Criticism.....	18
Customer Feedback.....	18

D

Declarations and nesting of the function blocks.....	49, 55
Devices.....	95
Documentation structure.....	6
Documentation:	
Record of revision.....	5

E

Energy.....	101
Energy consumption of ancillary components....	81
Energy consumption of drives.....	80
Energy saving functions.....	88
EnergyControl.....	25, 36
Architecture.....	27
Coupling.....	29
Functionality.....	27
Functioning.....	28
Modularization.....	28
Motivation.....	25
Next steps.....	26
Overview.....	210
Sercos Energy.....	26
EnergyFunctions.....	97
EnergyManagement.....	21, 85
Overview.....	21, 85
Properties.....	85
Purpose.....	85
EnergyManagement properties.....	85
EnergyMonitor.....	79, 92
Commissioning.....	79
Components.....	79
Energy consumption, ancillary components.....	81
Energy consumption, drives.....	80
Energy consumption, total.....	79
Overview.....	79
Properties.....	79
Start.....	79
EnergyMonitor block	
Configuration.....	50
EnergyMonitor properties.....	79
EnergyMonitoring.....	23, 35, 99, 111
Architecture.....	24
Configuration.....	50
Energy.....	101
Filtering the consumers.....	104
Functionality.....	24
Functioning.....	25
Language switching.....	105
Motivation.....	23
Next steps.....	23
Overview.....	35, 49, 99, 111
Performance.....	103
Realization.....	49

Index

Error codes

IL_4EEAdvancedDeviceType01.....	255
IL_4EEBasicCommandType01.....	228
IL_4EEBasicDeviceType01.....	248
IL_4EEEnergyControlType01.....	221
IL_4EEEnergyFunctionType01.....	257
IL_4EEEnergyMonitorType01.....	115
IL_4EEEventCommandType01.....	235
IL_4EEEventCommandType02.....	242
IL_4EEHMIDataCollectorType01.....	259
IL_4EELoadProfileEstimation.....	183
IL_4EEMonBasicDeviceType01.....	120
IL_4EEMonCompAirDeviceType01.....	153
IL_4EEMonFluidDeviceType01.....	158
IL_4EEMonGasDeviceType01.....	163
IL_4EEMonGasDeviceType02.....	168
IL_4EEMonHMIDataCollectorType01.....	186
IL_4EEMonHMIDataCollectorType02.....	188
IL_4EEMonHMIDataCollectorType03.....	190
IL_4EEMonIndraDriveDeviceType01.....	124
IL_4EEMonPneumaticDeviceType01.....	174
IL_4EEMonPowerSupplyDeviceType01.....	137
IL_4EEMonProfiEnergyDeviceType01.....	133
IL_4EEMonS0DeviceType01.....	178
IL_4EEMonScadaDataCollectorType01.....	194
IL_4EEMonSercosEnergyDeviceType01....	128, 286
IL_4EEMonTerminalDeviceType01.....	142
IL_4EEMonTerminalDeviceType02.....	148
IL_4EEScadaDataCollectorType01.....	263
Events.....	87, 96

F

Feedback.....	18
Filtering the consumers.....	104
Function blocks and their methods.....	113

H

Handling.....	37
Handling and press.....	35
Hazard warnings.....	6
Helpdesk.....	299
HMIDataCollector block	
Configuration.....	50
Hotline.....	299

I

IL_4EE_CHANGE_EM.....	269
IL_4EE_CMD_STATE.....	269
IL_4EE_CONFIG_ADVANCED_DE- VICE_EM_TYPE01.....	267
IL_4EE_CONFIG_ADVANCED_DE- VICE_TYPE01.....	266
IL_4EE_CONFIG_BASIC_COM- MAND_TYPE01.....	268
IL_4EE_CONFIG_BASIC_DE- VICE_EM_TYPE01.....	266

IL_4EE_CONFIG_BASIC_DEVICE_TYPE01....	265
IL_4EE_CONFIG_ENERGY_CON- TROL_EM_TYPE01.....	264
IL_4EE_CONFIG_ENERGY_CON- TROL_TYPE01.....	263
IL_4EE_CONFIG_ENERGY_FUNC- TION_TYPE01.....	268
IL_4EE_CONFIG_EVENT_COM- MAND_TYPE01.....	265
IL_4EE_CYLINDER_PARAMETER.....	200
IL_4EE_DEVICE_CONFIG_DE- VICE_TYPES.....	202
IL_4EE_DURATION_TIME.....	205
IL_4EE_EM_ASSIGNMENT_TYPE01.....	268
IL_4EE_ENERGY_COMPONENT_MIN.....	205
IL_4EE_ENERGY_CONTROL_STATE.....	296
IL_4EE_ENERGY_DATA_TYPE20_MIN.....	204
IL_4EE_ENERGY_DATA_TYPE64_MIN.....	203
IL_4EE_FLUID_MATERIAL_PARAMETER.....	199
IL_4EE_GUI_MON_CONFIG.....	201
IL_4EE_GUI_MON_DEVICE_CONFIG.....	201
IL_4EE_GUI_MON_RESULT_MIN.....	202
IL_4EE_GUI_MON_RESULT_TYPE20_MIN....	203
IL_4EE_GUI_MON_RESULT_TYPE64_MIN....	203
IL_4EE_HMI_COMMAND_INIT_TYPE01.....	273
IL_4EE_HMI_COMMAND_TYPE01.....	273
IL_4EE_HMI_COMMAND_UP- DATE_TYPE01.....	274
IL_4EE_HMI_CONSTANTS_TYPE01.....	271
IL_4EE_HMI_CONTROL_EM_TYPE01.....	271
IL_4EE_HMI_CONTROL_INIT_TYPE01.....	272
IL_4EE_HMI_CONTROL_TYPE01.....	271
IL_4EE_HMI_CONTROL_UP- DATE_TYPE01.....	272
IL_4EE_HMI_DATA_TYPE01.....	269
IL_4EE_HMI_DEVICE_EM_TYPE01.....	275
IL_4EE_HMI_DEVICE_INIT_TYPE01.....	274
IL_4EE_HMI_DEVICE_TYPE01.....	274
IL_4EE_HMI_DEVICE_UPDATE_TYPE01.....	275
IL_4EE_HMI_ENERGY_FUNCTION_IN- IT_TYPE01.....	276
IL_4EE_HMI_ENERGY_FUNC- TION_TYPE01.....	275
IL_4EE_HMI_ENERGY_FUNCTION_UP- DATE_TYPE01.....	276
IL_4EE_MON_DEVICE_TYPE.....	194
IL_4EE_MON_ENERGY_TYPE.....	194
IL_4EE_MON_FLOWRATE_UNIT(ENUM).....	199
IL_4EE_MON_FLUID_MATERIAL_TYPE.....	199
IL_4EE_MON_GAS_TYPE(ENUM).....	200
IL_4EE_MON_HMI_CONSTANTS_TYPE02.....	206
IL_4EE_MON_HMI_CONSTANTS_TYPE03.....	208
IL_4EE_MON_HMI_DEVICES_DA- TA_TYPE02.....	206
IL_4EE_MON_HMI_DEVICES_DA- TA_TYPE03.....	209
IL_4EE_MON_HMI_DEVICES_TYPE02.....	206
IL_4EE_MON_HMI_DEVICES_TYPE03.....	209

IL_4EE_MON_HMI_MEASUREMENT_RE-	
SULT_TYPE02.....	207
IL_4EE_MON_HMI_MEASUREMENT_RE-	
SULT_TYPE03.....	210
IL_4EE_MON_HMI_MEASUREMENT_STA-	
TUS.....	210
IL_4EE_MON_HMI_MEASURE-	
MENT_TYPE02.....	207
IL_4EE_MON_HMI_MEASURE-	
MENT_TYPE03.....	209
IL_4EE_MON_HMI_MODE.....	210
IL_4EE_MON_HMI_STRINGS_TYPE02.....	207
IL_4EE_MON_HMI_TYPE02.....	206
IL_4EE_MON_HMI_TYPE03.....	208
IL_4EE_MON_PRESSURE_UNIT.....	199
IL_4EE_MON_PROFI_ENERGY_AVAILABI-	
TY.....	195
IL_4EE_MON_TEMPERA-	
TURE_UNIT(ENUM).....	200
IL_4EE_MON_TERMINAL_BASIC_VALUES....	196
IL_4EE_MON_TERMINAL_ENERGY_VAL-	
UES.....	198
IL_4EE_MON_TERMINAL_EXTENDED_VAL-	
UES.....	196
IL_4EE_MON_TERMINAL_IN_PROC_DATA...	196
IL_4EE_MON_TERMINAL_POWER_VAL-	
UES.....	197
IL_4EE_MON_TERMINAL_REAC-	
TIVE_POWER_VALUES.....	198
IL_4EE_POWER_COMPONENT_MIN.....	205
IL_4EE_POWER_DATA_TYPE20_MIN.....	204
IL_4EE_POWER_DATA_TYPE64_MIN.....	204
IL_4EE_PROFI_ENERGY_END_PAUSE.....	287
IL_4EE_PROFI_ENERGY_GET_MEASURE-	
MENT_LIST.....	289
IL_4EE_PROFI_ENERGY_GET_MEASURE-	
MENT_VALUES.....	290
IL_4EE_PROFI_ENERGY_GET_MODE.....	287
IL_4EE_PROFI_ENERGY_IDENTIFY.....	289
IL_4EE_PROFI_ENERGY_LIST_ENER-	
GY_SAVING_MODES.....	287
IL_4EE_PROFI_ENERGY_MEAS_VALUE.....	290
IL_4EE_PROFI_ENERGY_START_PAUSE.....	287
IL_4EE_PROFI_ENERGY_STATUS.....	288
IL_4EE_RINGBUFFER_MODE.....	296
IL_4EE_TIMESTAMP_MIN.....	204
IL_4EE_TYPE_SEL.....	269
IL_4EEAdvancedDeviceType01.....	248
IL_4EEBasicCommandType01.....	221
IL_4EEBasicDeviceType01.....	242
IL_4EEEnergyControlType01.....	212
IL_4EEEnergyFunctionType01.....	255
IL_4EEEnergyMonitorType01.....	113
IL_4EEEventCommandType01.....	228
IL_4EEEventCommandType02.....	235
IL_4EEHMIDataCollectorType01.....	257
IL_4EELoadProfileEstimation.....	178
IL_4EEMonBasicDeviceType01.....	115

IL_4EEMonCompAirDeviceType01.....	148
IL_4EEMonFluidDeviceType01.....	153
IL_4EEMonGasDeviceType01.....	158
IL_4EEMonGasDeviceType02.....	163
IL_4EEMonHMIDataCollectorType01.....	183
IL_4EEMonHMIDataCollectorType02.....	186
IL_4EEMonHMIDataCollectorType03.....	188
IL_4EEMonIndraDriveDeviceType01.....	120
IL_4EEMonPneumaticDeviceType01.....	168
IL_4EEMonPowerSupplyDeviceType01.....	133
IL_4EEMonProfiEnergyDeviceType01.....	128
IL_4EEMonS0DeviceType01.....	174
IL_4EEMonScadaDataCollectorType01.....	190
IL_4EEMonSercosEnergyDeviceType01.....	124
IL_4EEMonTerminalDeviceType01.....	137
IL_4EEMonTerminalDeviceType02.....	142
IL_4EEProfiEnergyServiceType01.....	276
IL_4EEScadaDataCollectorType01.....	259
Initializing the consumers.....	71
Inserting .NET controls in WinStudio.....	89
Interfaces and operation of the controls.....	91
General information.....	91
Introduction.....	19

L

Language switching.....	105
Library	
RIL_4EEEnergyManagement.library.....	111

M

Machine.....	94
Machine information.....	86
MachineEnergyCtrlCmdControlType.....	295
MachineEnergyCtrlCmdStatusType.....	296
MachineEnergyCtrlConfigType.....	294
MachineEnergyCtrlStatusType.....	295
MachineEnergyCtrlType.....	293
MachineEnergyMonConsumptionType.....	292
MachineEnergyMonDevicesType.....	293
MachineEnergyMonRingbufferType.....	293
MachineEnergyMonType.....	291
MachineEnergyObjectType.....	290
mGetCommandState	
IL_4EEBasicCommandType01.....	223
IL_4EEEventCommandType01.....	230
IL_4EEEventCommandType02.....	237
mGetControl	
IL_4EEAdvancedDeviceType01.....	249
IL_4EEBasicDeviceType01.....	243
IL_4EEEnergyControlType01.....	214
mGetState	
IL_4EEAdvancedDeviceType01.....	250
IL_4EEBasicDeviceType01.....	244
IL_4EEEnergyControlType01.....	214
Modularization.....	28
Module: Handling.....	31
Module: Press.....	32

Index

MonitoringDevice block	
Configuration.....	50
mResetCommand	
IL_4EEBasicCommandType01.....	223
IL_4EEEventCommandType01.....	230
IL_4EEEventCommandType02.....	237
mSetAdditivePowerConsumption	
IL_4EEMonIndraDriveDeviceType01.....	122
IL_4EEMonPowerSupplyDeviceType01.....	135
mSetConsumerActive	
IL_4EEMonBasicDeviceType01.....	118
mSetDirectCommand	
IL_4EEBasicCommandType01.....	222
mSetEnergyOdometer	
IL_4EEMonBasicDeviceType01.....	117
mSetEventCommand	
IL_4EEEventCommandType01.....	229
IL_4EEEventCommandType02.....	236
mSetFlowMeter	
IL_4EEMonFluidDeviceType01.....	156
IL_4EEMonGasDeviceType01.....	161
mSetGasFlowMeter	
IL_4EEMonCompAirDeviceType01.....	150
mSetPauseCommand	
IL_4EEBasicCommandType01.....	222
mSetPowerConsumption	
IL_4EEMonBasicDeviceType01.....	117
mSetPressure	
IL_4EEMonCompAirDeviceType01.....	151
mSetS0Signal	
IL_4EEMonGasDeviceType02.....	165
IL_4EEMonS0DeviceType01.....	176
mSetState	
IL_4EEAdvancedDeviceType01.....	250
IL_4EEBasicDeviceType01.....	244
IL_4EEEnergyControlType01.....	214
mSetValue	
IL_4EEEnergyFunctionType01.....	256

N

Names.....	17
Names and abbreviations.....	17

O

Output power of the ancillary components.....	84
Output power of the drives.....	83
Overview.....	99

P

Parameter list.....	297
Performance.....	103
PowerMonitor.....	81, 93
Commissioning.....	82
Components.....	81
Configuration, general.....	82
Output power, components.....	84
Output power, drives.....	83

Overall power.....	83
Overview.....	81
Properties.....	81
Purpose.....	81
Start.....	82
Zoom.....	84
PowerMonitor properties.....	81
Press.....	42
Purpose of EnergyManagement.....	85
Purpose of EnergyMonitor.....	79
Purpose of PowerMonitor.....	81

R

Realizing EnergyControl.....	55
Overview.....	55
Realizing the EnergyMonitoring.....	49
RIL_4EEEnergyManagement.library.....	111
Overview.....	111

S

Safety.....	37
Safety instructions.....	6
Sercos Energy.....	26
Service hotline.....	299
Setting the energy modes for machine and stations.....	36
Signal alert symbol.....	6
Signal words.....	6
Specifications and commands.....	37
Specifying dead time or energy mode.....	85
Starting the EnergyMonitor.....	79
Suggestions.....	18
Support.....	299
Switching consumers after initialization.....	73
Symbols.....	8

T

Target group.....	5
Total energy consumption.....	79

U

Using commissioning visualization.....	52, 74

V

Visualizing SmartAutomation.....	76

W

Warnings.....	6
Web-based visualization.....	98
Overview.....	98
WinStudio.....	88
Overview.....	88
Working with the EnergyManagement.....	86
Working with the EnergyManagement (dead time or energy mode).....	85
Requirements.....	85
Working with the EnergyMonitor.....	79

Index

Working with the PowerMonitor..... 82

Z

Zoom..... 84

Notes

Notes

Bosch Rexroth AG

Electric Drives and Controls

P.O. Box 13 57

97803 Lohr, Germany

Bgm.-Dr.-Nebel-Str. 2

97816 Lohr, Germany

Phone +49 9352 18 0

Fax +49 9352 18 8400

www.boschrexroth.com/electrics



R911339229