



Programming manual

edgeController

AE1300

English

Contents

1	Preliminary note	4
1.1	Legal and copyright information	4
1.2	Explanation of Symbols	5
1.3	Change history	5
2	Safety instructions	6
2.1	General safety instructions	6
2.2	Required background knowledge	6
3	System requirements	7
3.1	Hardware	7
3.2	Software	7
3.3	Licensing	7
4	Installation	8
4.1	CODESYS Development System	8
4.1.1	Install CODESYS Development System	8
4.2	CODESYS package for edgeController	9
4.2.1	Components of the package	9
4.2.2	Installing the package	9
4.2.3	Updating the package	10
4.2.4	Uninstalling the package	10
5	Getting started	11
5.1	Creating a CODESYS project	11
5.2	Project structure	12
6	System configuration	13
6.1	Configuring the programming interface	13
6.1.1	Setting the communication path	13
6.2	Configuring the Ethernet ports of the device	14
6.3	Securing the project	15
6.4	Installing IODDs	15
7	Programming	16
7.1	Objects of the application (project template)	16
7.2	Creating an application	17
7.3	Notes	17
7.3.1	ifm function libraries	17
7.3.2	Supported programming languages	17
7.3.3	Available memory	17
7.3.4	Supported variable types	18
7.3.5	Using sample applications	18
7.3.6	Device temperature	18
7.4	IoT Core	20
7.4.1	Notes: ifm IoT Core	20
7.4.2	Example application: IoT Core	20
7.4.3	Instantiating auxiliary functions	21

7.4.4	Onboarding devices	22
7.4.5	Managing devices	23
7.4.6	Manual creation of data points in the ifm IoT Core	23
7.4.7	Reading or writing data points of the ifm IoT Core	24
7.4.8	Tagging data points for cloud systems	24
7.4.9	Enabling the OPC-UA server.....	25
7.4.10	Transferring data to the cloud.....	25
7.5	Cloud systems.....	26
7.5.1	Example application: clouds.....	26
7.5.2	Function elements	26
7.5.3	Configuring the cloud adapter	27
7.5.4	Establishing the connection to the cloud systems	30
7.6	Visualisation	31
7.6.1	Creating a new visualisation page.....	31
7.6.2	Supported visualisation types.....	31
7.6.3	Visualisation tasks.....	31
7.6.4	Use image collection	32
7.6.5	Example application: visualisation.....	32
7.6.6	Visualising cloud operations.....	32
7.6.7	Setting the properties of the visualisation	33
7.7	Task configuration.....	33
7.7.1	Setting the task configuration	33
8	Operation	34
8.1	Transfer CODESYS project to the device.....	35
8.1.1	Load application to AE1300.....	35
8.1.2	Delete application from the device	36
8.2	Operating states of the PLC application	37
8.2.1	Display operating mode of the PLC application.....	37
8.2.2	Start PLC application.....	37
8.2.3	Stop PLC application.....	37
8.3	Reset.....	38
8.3.1	Supported reset variants	38
8.3.2	Reset application (warm).....	38
8.3.3	Reset application (cold).....	39
8.3.4	Reset application (origin).....	39
8.4	Display system information	40
9	Appendix	41
9.1	Library: ifm_iot.library.....	42
9.1.1	IoT Core	43
9.1.2	HMI_CloudConnect.....	70
9.1.3	Cloud.....	76
9.2	Library ifmDevice_ecomatDisplay.library.....	79
9.2.1	GetLightSensor	80
9.2.2	etLightSensorCyclic.....	82
9.2.3	GetTemperature	84
9.2.4	GetTemperaturesCyclic.....	86
9.2.5	LCDcontrol	88
9.2.6	GetBootupBacklight.....	90
9.2.7	GetLCD_Backlight.....	92
9.2.8	SetBootupBacklight.....	94
9.2.9	SetLCD_Backlight	96

1 Preliminary note

Content

Legal and copyright information	4
Explanation of Symbols	5
Change history	5

33203

1.1 Legal and copyright information

33117

© All rights reserved by ifm electronic gmbh. No part of this manual may be reproduced and used without the consent of ifm electronic gmbh.

All product names, pictures, companies or other brands used on our pages are the property of the respective rights owners:

- AS-i is the property of the AS-International Association, (→ www.as-interface.net)
- CAN is the property of the CiA (CAN in Automation e.V.), Germany (→ www.can-cia.org)
- CODESYS™ is the property of the CODESYS GmbH, Germany (→ www.codesys.com)
- DeviceNet™ is the property of the ODVA™ (Open DeviceNet Vendor Association), USA (→ www.odva.org)
- EtherNet/IP® is the property of the → ODVA™
- EtherCAT® is a registered trade mark and patented technology, licensed by Beckhoff Automation GmbH, Germany
- IO-Link® is the property of the → PROFIBUS Nutzerorganisation e.V., Germany (→ www.io-link.com)
- ISOBUS is the property of the AEF – Agricultural Industry Electronics Foundation e.V., Deutschland (→ www.aef-online.org)
- Microsoft® is the property of the Microsoft Corporation, USA (→ www.microsoft.com)
- Modbus® is the property of the Schneider Electric SE, France (→ www.schneider-electric.com)
- PROFIBUS® is the property of the PROFIBUS Nutzerorganisation e.V., Germany (→ www.profibus.com)
- PROFINET® is the property of the → PROFIBUS Nutzerorganisation e.V., Germany
- Windows® is the property of the → Microsoft Corporation, USA

1.2 Explanation of Symbols

34171



WARNING

Warning of serious personal injury.
Death or serious irreversible injuries may result.



CAUTION

Warning of personal injury.
Slight reversible injuries may result.



NOTICE

Warning of damage to property



Important note
Non-compliance can result in malfunction or interference



Information
Supplementary note



Request for action



Reaction, result



"see"

abc

Cross-reference

123

Decimal number

0x123

Hexadecimal number

0b010

Binary number

[...]

Designation of pushbuttons, buttons or indications

1.3 Modification history

61690

Version	Topic	Date
00	New creation of the document	05 / 2021

2 Safety instructions

Content

General safety instructions	6
Required background knowledge	6

28333

2.1 General safety instructions

41415

Read this document before setting up the product and keep it during the entire service life.

Only use the product for its intended purpose.

If the operating instructions or the technical data are not adhered to, personal injury and/or damage to property may occur.

Improper or non-intended use may lead to malfunctions of the device, to unwanted effects in the application or to a loss of the warranty claims.

The manufacturer assumes no liability for any consequences caused by tampering with the device or incorrect use by the operator.

- Observe these operating instructions.
- Adhere to the warning notes on the product.

2.2 Required background knowledge

41648

This document is intended for specialists. Specialists are people who, based on their relevant training and experience, are capable of identifying risks and avoiding potential hazards that may be caused during operation or maintenance of the product.

For programming these people should also have knowledge of control technology experience in PLC programming to IEC 61131-3.

The document contains information about the correct handling of the product.

3 System requirements

Content

Hardware	7
Software.....	7
Licensing.....	7

38972

3.1 Hardware

61691

- PC for CODESYS programming system (hardware requirements: see CODESYS help)
- edgeController AE1300, Firmware: or higher
- IO-Link Master DataLine AL13nn / CabinetLine AL19nn, firmware: 3.1 or higher
- IO-Link Device (sensors /actuators)
- Connection technology
 - Ethernet connection cable
 - Connection cable IO-Link devices

3.2 Software

61692I

- CODESYS Development System, Version V3.5 SP15 Patch 1 (64 bit)
- Package ""ifm edgeController with display"", Version 1.0.1.2
- When using the Microsoft Azure cloud system: Azure IoT Explorer (→ github.com/Azure/azure-iot-explorer/releases)



The features and functions guaranteed in this manual can only be achieved with the software components in the versions specified here! ifm electronic offers the software components for download on the website: → www.ifm.com

3.3 Licensing

43510

By purchasing the AE1300, the user also acquires a valid licence to use the "ifm edgeController with display".



Licence information: → Installation routine or product page of the article on the ifm website.

4 Installation

Content

CODESYS Development System	8
CODESYS package for edgeController	9

42637

4.1 CODESYS Development System

39481

The CODESYS Development System (short: CODESYS) is a platform for the creation of PLC applications according to the standard IEC 61131-3.

4.1.1 Install CODESYS Development System

40432

To install the software "CODESYS Development System":

- ▶ Install the programming system CODESYS V3.5 SP15 Patch 1 (→ CODESYS installation and first steps).
- > CODESYS V3.5 SP15 Patch 1 is installed on the programming PC/laptop.

4.2 CODESYS package for edgeController

Content

Components of the package.....	9
Installing the package	9
Updating the package.....	10
Uninstalling the package	10

61693

4.2.1 Components of the package

61694

ifm electronic provides a package to program the edgeController with the CODESYS development environment. The package contains the following components:

- Device descriptions
- Function libraries
- Project templates with application examples



Documentation of the ifm function libraries: **Library: ifm_iot.library** (→ p. [42](#))

4.2.2 Installing the package

61695

Requirements:

- CODESYS V3.5 SP15 Patch 1 has been installed correctly.
- The file "ifm_edgeController_64bit_Vn.n.n.n.package" has been downloaded and unpacked on the PC.
- ▶ Start CODESYS with administrator rights.
- ▶ Select [Tools] > [Package Manager].
- > Package Manager appears.
- ▶ Click on [Install...].
- > The file explorer is displayed.
- ▶ Select the file "ifm_edgeController_64bit_Vn.n.n.n.package".
- ▶ Click on [Open] and execute a complete installation.
- > The package "ifm edgeController with display" will be installed.

After successful installation:

- > Package Manager shows installed package.
- > The functions of the package can be used.
- ▶ Click [Close] to quit the Package Manager.

4.2.3 Updating the package

61696

Requirements:

- CODESYS has been started with administrator rights.
- ▶ Uninstall the outdated version of the package (→ **Uninstalling the package** (→ p. [10](#))).
- ▶ Install a new version of the package (→ **Installing the package** (→ p. [9](#))).
- ▶ Open the project.
- ▶ In the device tree: Select the node of the device
- ▶ Select [Project] > [Update Device...].
- > A dialogue window appears.
- ▶ Click on [Update Device] to start the update process.
- > CODESYS loads new device libraries.
- > Device tree view is updated.
- ▶ Click on [Close] to close the dialogue window.
- ▶ Save the project.

4.2.4 Uninstalling the package

61697

Requirements:

- CODESYS has been started with administrator rights.
- ▶ Select [Tools] > [Package Manager...].
- > Package Manager appears.
- > The Package Manager shows the installed packages.
- ▶ Select the package to be uninstalled.
- ▶ Click on [Uninstall...].
- > CODESYS uninstalls the selected package.
- ▶ Click [Close] to quit the Package Manager.

5 Getting started

Content

Creating a CODESYS project.....	11
Project structure	12

61698

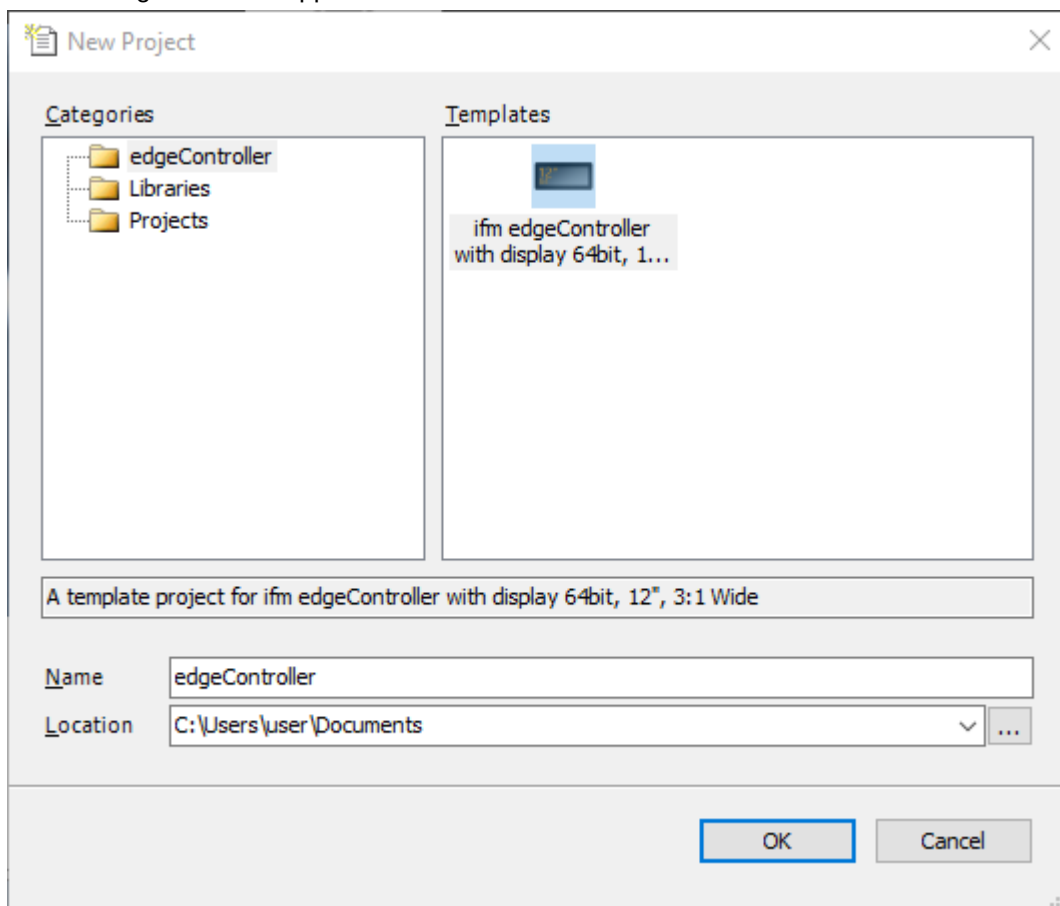
5.1 Creating a CODESYS project

61699

The package contains a template to create a CODESYS project with a AE1300.

Requirements:

- All necessary components have been installed without errors (→ **Installation** (→ p. 8)).
- ▶ Start CODESYS Development System
- ▶ Select [File] > [New Project].
- > A dialogue window appears.

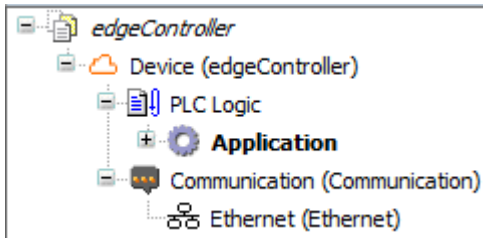


- ▶ Select the [edgeController] in [Categories].
- ▶ Select the required template in [Templates].
- ▶ Enter a name for the project and select a storage location.
- ▶ Click on [OK].
- > CODESYS creates project with edgeController.
- > All necessary function libraries have been integrated into the project.

5.2 Project structure

61701

A CODESYS project contains all components to configure, manage and program the ecomatDisplay. All components of a project are shown in the window [Devices] in a hierarchic tree view. CODESYS projects with a edgeController have the following structure:



Legend:

- edgeController logical root node of the device, provides access to the general settings of the device
- PLC logic includes the application of the device
- Application includes the objects that are required for a control program and the visualisation.
- Communication provides access to the configuration options of the communication interfaces
- Ethernet available communication interfaces.

6 System configuration

Content

Configuring the programming interface	13
Configuring the Ethernet ports of the device	14
Securing the project.....	15
Installing IODDs.....	15

61703

6.1 Configuring the programming interface

61704

To be able to load an application onto the device, the connection between CODESYS and the device must be configured and activated.

The device-internal PLC can be programmed via one of the Ethernet interfaces of the device (position of the connections: → Operating instructions)



The device and the CODESYS PC can be coupled directly or indirectly via an Ethernet network.

- ▶ Only use the recommended accessories for connection of the Ethernet interfaces (→ Operating instructions)!
- ▶ If necessary, configure the Ethernet interface of the device in the setup mode (→ Operating instructions).

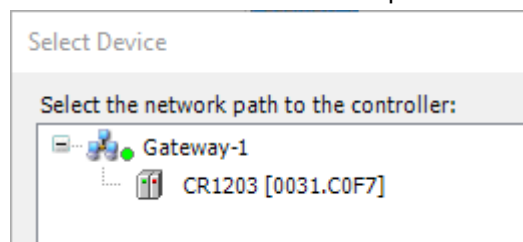
6.1.1 Setting the communication path

61705

To set the communication path between the CODESYS programming system and the unit:

Requirements:

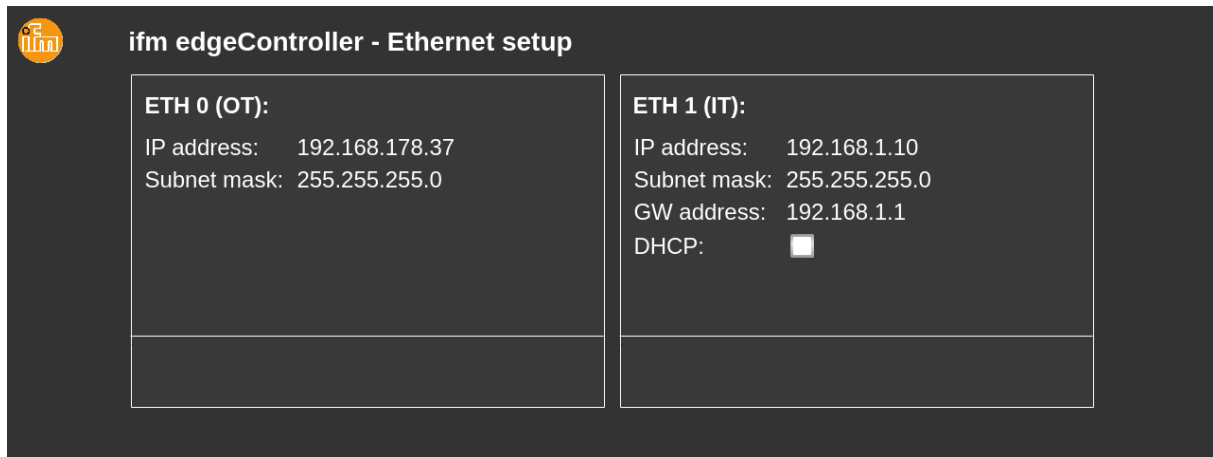
- The CODESYS PC and device are connected via Ethernet interfaces.
- The Ethernet interfaces are configured.
- ▶ edgeController has been started and is in runtime mode (application has been started).
- ▶ In CODESYS: Open the project.
- ▶ In the device tree: Double-click on the [edgeController] symbol.
- ▶ In the editor window: Select the [Communication] tab.
- > Editor window shows communication settings.
- ▶ Select the requested gateway in the list [Gateway].
- ▶ Enable the [Sacn Network...] button.
- > The [Select Device] button appears.
- > CODESYS scans the network for devices.
- > Window shows the network path and detected devices.



- ▶ Select node of the device and click on [OK] to set the communication path.
- > CODESYS can transfer data to the device-internal PLC.

6.2 Configuring the Ethernet ports of the device

Upon delivery, a demo application is stored on the device. The application allows the user to configure the Ethernet interfaces of the unit (→ Operating instructions).



ifm edgeController - Ethernet setup

ETH 0 (OT):	ETH 1 (IT):
IP address: 192.168.178.37	IP address: 192.168.1.10
Subnet mask: 255.255.255.0	Subnet mask: 255.255.255.0
	GW address: 192.168.1.1
	DHCP: ☐

► Use the demo application to configure the IP settings of the Ethernet ports.

The demo application is also part of the project template.

Components of the demo application:

Component	Description
IPSetting (PRG)	Program code
Visu_Eth_Settings	Visualisation page; contains the operating and display elements of the
ifmImagePool	Image collection; contains the ifm logo shown

To test the connection to the unit:

Requirements:

- The programming interface has been configured.
- The CODESYS PC and edgeController are connected via Ethernet interfaces.
- edgeController is switched on.
- The demo application will be processed.
- In CODESYS: Select [Online] > [Login].
- > CODESYS complies with the program code.
- > CODESYS transfers the compiled code to the device.
- Select [Debug] > [Start].
- > The application on the device will be started.
- > The display shows the visualisation of the demo application.

6.3 Securing the project

61708

CODESYS offers the following security functions:

CODESYS Development System

- Encryption of the source code of the application
- User administration at project level
- Encrypted communication between the CODESYS Development System and the controller:

CODESYS runtime system

- User administration for access to the control system
- Encryption and signing of the executable application code
- Operating modes for the executable application code
- Interactive login to the target device
- Easy replacement or restoration of controls
- Encrypted OPC UA communication

Application code

- Access restrictions by the application
- Unlocking additional functions

Visualisation

- User administration for visualisations
- Encrypted communication for CODESYS WebVisu



Take appropriate measures to protect the device and communication from unauthorised access!

6.4 Installing IODDs

61709

To enable communication of the edgeController with the IO-Link devices, the device descriptions (IO Device Descriptions - IODD) of the IO-Link sensors and IO-Link actuators must be installed on the device. IODD files can be downloaded free of charge from the website → ioddfinder.io-link.com.

Requirements:

- The programming interface has been configured.
- IODDs of the IO-Link devices have been downloaded.
- CODESYS is logged in on edgeController.
- ▶ In the device tree: Double-click on the [edgeController] symbol.
- ▶ In the editor window: Select the [Files] tab.
- > The editor window shows the file manager.
- ▶ Transfer IODDs to the following directory on the edgeController:
/usr/share/ifm/Iodds
- > edgeController can communicate with IO-Link devices.

7 Programming

Content

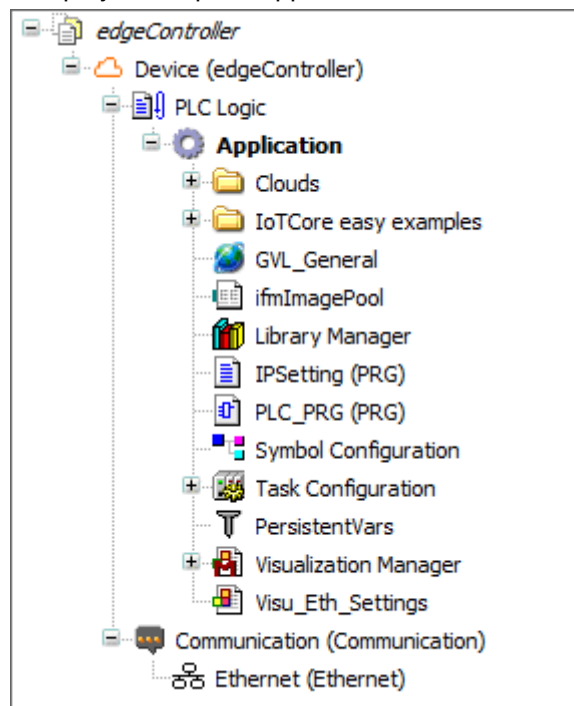
Objects of the application (project template)	16
Creating an application	17
Notes	17
IoT Core	20
Cloud systems	26
Visualisation	31
Task configuration	33

61710

7.1 Objects of the application (project template)

61711

All objects of a PLC application are listed as subelements of the node [Application] in the device tree. The project template application includes the following objects:



Legend:

- Clouds Access to program blocks for connection to supported cloud systems
- IoTCore easy examples Access to program block examples (IoT core functions, data transfer to cloud)
- GVL_General Access to Global Variables of the project
- Library manager Access to function libraries of the project
- IPSetting (PRG) Program code of the start application
- PLC_PRG (PRG) Access to the main program; contains the program code and calls to the sub-programs
- Symbol configuration Access to OPC-UA function
- Task configuration Access to the properties of the task processing
- PersistentVars Persistent variables to subscribe data
- Visualisation Manager Access to the properties of the visualisations:
- Visu_Eth_Settings Visualisati→ p. of the start application

7.2 Creating an application

61713

CODESYS automatically generates the function block PLC_PRG (PRG) during project creation. The function block is processed cyclically. Sub-programs will be called in this function block.

To create a PLC application:

- ▶ In the device tree: Double-click on [Application] > [PLC_PRG (PRG)]
- > The editor window shows the input mask of the selected programming language.
- ▶ Enter program code.

7.3 Notes

36782

7.3.1 ifm function libraries

61714

The package includes function libraries to program the AE1300. The function libraries created by ifm are documented in this document (→ **Appendix** (→ p. [41](#))).

Function libraries from other providers contain their own descriptions (→ Library manager).

7.3.2 Supported programming languages

57733

The following programming languages according to IEC 61131 are supported by the ifm function libraries:

- Function Block Diagram FBD
- Ladder Diagram LD
- Structured Text ST
- Sequential Function Chart SFC
- Instruction List IL
- Continuous Function Chart CFC

7.3.3 Available memory

61715

Memory area	Size
Flash memory	8 GB
RAM memory	1 GB

7.3.4 Supported variable types

61716

The device supports the following variable types:

Variable type	Declaration	Scope of validity	Memory behaviour
local	In the declaration part of the POU	Applies only to the POU in which it has been declared	volatile
global	In the global variable list (GVL)	Applies to all POUs of the project	volatile
global retain			non volatile
Network	In network variable lists	Values are available to all projects in the whole network if the variable is contained in their network variables lists.	volatile

7.3.5 Using sample applications

61717

The project template includes the following sample programs:

- PLC_PRG_IoTCore (PRG): IoT Core functionality
- FB_SubscribeDataFromIoTCore (FB): Cloud connection



The programme modules of the template include comments. The comments will only be visible if the display function for comments is enabled.

- ▶ In the project: Select [Tools] > [Options].
- > A dialogue window appears.
- ▶ Select [FUP, FUP and AWL] in the left column.
- ▶ In the [View] area, enable the options [Show network title] and [Show network comment].
- ▶ Click on [Add] to save the changes.

7.3.6 Device temperature

61276

The internal temperature sensor and the light sensor of the device can be read out with the software.

The device primarily heats up due to the following factors:

- Solar radiation
- Ambient temperature
- Installation conditions
- External heating
- Current consumption (display illumination etc.)

The device heats up considerably under demanding operating conditions. The device temperature must not exceed the maximum permissible value of 80°C.

The values provided by the internal temperature sensor can e.g. be used for the following actions:

- ▶ Comply with the maximum permissible ambient temperature
- ▶ Reducing the display brightness to decrease the self-heating
- ▶ Protect the unit from external heat (e.g. from other units).
- ▶ Warning the operator about heating and an upcoming switch-off of the display
- ▶ Switch-off of the display

NOTICE!

Internal temperature exceeds the permissible value (→ Data sheet)

- > Malfunction or damage to the device possible
- ▶ Reduce device temperature.
- ▶ Take appropriate measures to prevent further heating up of the device.

7.4 IoT Core

61718

7.4.1 Notes: ifm IoT Core

61719

The ifm IoT Core represents the functionality of a device. The ifm IoT Core makes the data, services and events of a device available to the outside world via standard interfaces (http(s), OPC, MQTT, TCP). This allows people and other devices to access this information (parameters, process data, events) and functions (services).

The elements of the ifm IoT Core are arranged in a hierarchical tree structure in a JSON object. Each device is represented by a number of objects, services and events. An element is accessed via its address (e.g. root/port1/pin2). The address is composed of the path leading to the element (root/port1) and the identifier of the element (pin2).

The ifm IoT Core makes it possible to integrate the functionality of one device into the ifm IoT Core of another device. At the same time, the structure of one device is mirrored in the ifm IoT Core of the other device. This process is called "onboarding".

Example:

Device 1 (e.g. edgeController)

```
device1/getidentity
device1/gettree
device1/somedata
device1/somedata/getdata
```

Device 2 (e.g. IO-Link master)

```
device2/getidentity
device2/gettree
device2/somedata
device2/someotherdata/getdata
```

After integration of device 2 into device 1:

```
device1/getidentity
device1/gettree
device1/somedata
device1/somedata/getdata
device1/remote/device2/getidentity
device1/remote/device2/gettree
device1/remote/device2/somedata
device1/remote/device2/someotherdata/getdata
```

This makes it possible to access the data and services of device 2 via device 1.

7.4.2 Example application: IoT Core

61720

The project template includes the following program blocks in the [IoTCore easy examples] folder:

Name	Description
PLC_PRG_IoTCore (PRG)	implementation example of the most important functions to manage the ifm IoT Core
FB_SubscribeDataFromIoTCore (FB)	implementation example of the data transfer to a cloud system

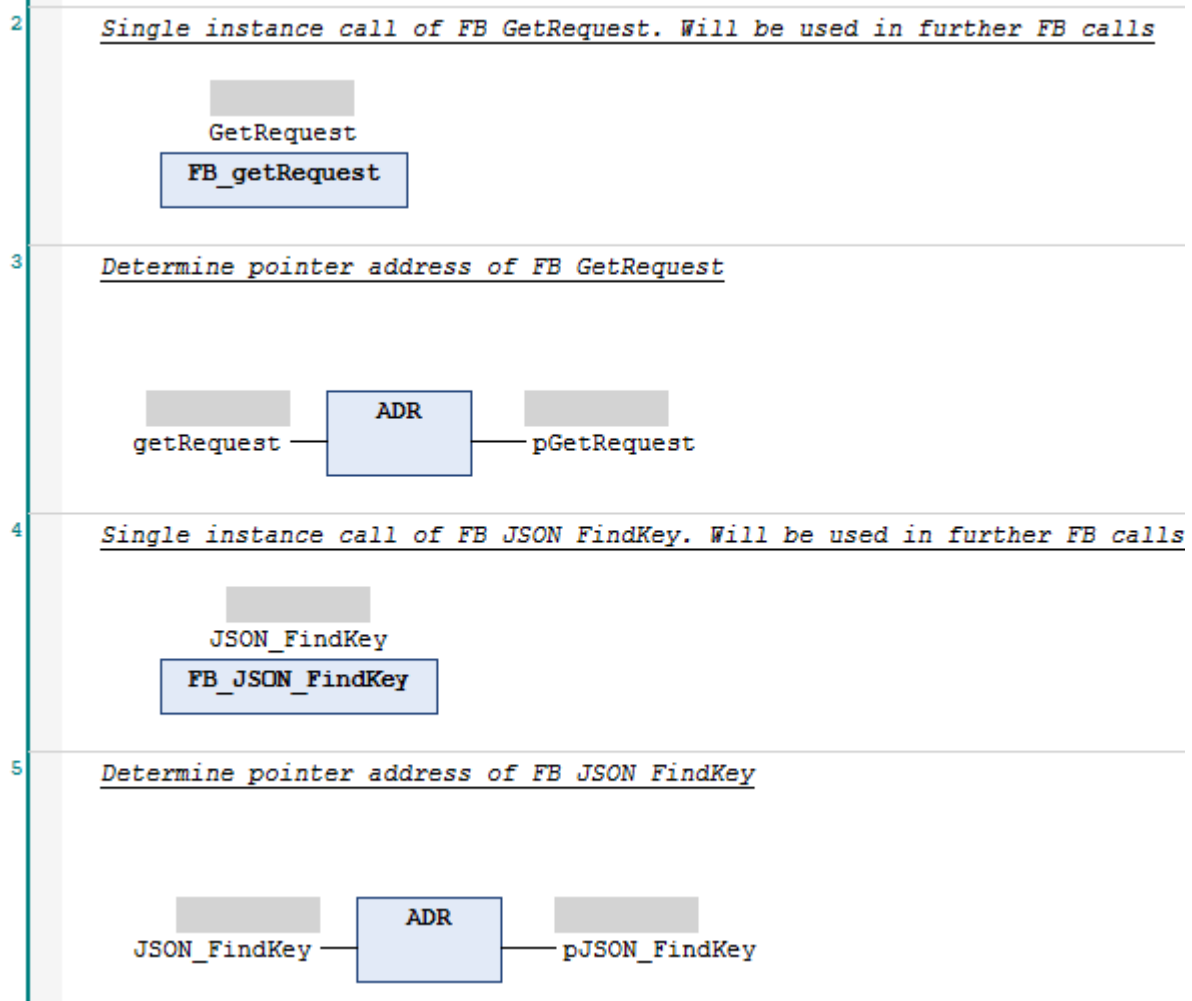


There are comments to the program blocks.

► Familiarise yourself with the program block functions!

7.4.3 Instantiating auxiliary functions

The elements of the "ifmIoT.lib" library use the functionality of the function blocks "FB_GetRequest" and "FB_JSON_FindKey". Therefore, one instance of each FB must be called in the application, and then the pointer address must be determined.



- ▶ Call the function block "FB_GetRequest".
- ▶ Use the function block "ADR" to determine the pointer address of the function block "FB_GetRequest".
- ▶ Call the function block "FB_JSON_FindKey".
- ▶ Use the function block "ADR" to find the pointer address of the function block "FB_JSON_FindKey".

7.4.4 Onboarding devices

61722

To access the process data of sensors and actuators connected to an ifm IoT Core enabled device, the ifm IoT Core enabled device must be mirrored in the edgeController (**Notes: ifm IoT Core** (→ p. 20)).

Available function blocks:

Name	Description
FB_onboard (FB) (→ p. 65)	The function block adds an ifm IoT Core enabled device to the ifm IoT Core of the edgeController (onboarding).

Requirements:

- Sensors / actuators have been configured.
- Sensors / actuators are connected to an ifm IoT Core enabled device.
- ifm IoT Core-capable device is connected to edgeController via the Ethernet network.
- For IO-Link devices: IODDs of the sensors / actuators are available on edgeController (**Installing IODDs** (→ p. 15)).
- ▶ Onboard the ifm IoT Core-capable device.
- > The device is mirrored in the IoT Core of the edgeController.
- > All mirrored data points are updated at the selected interval.
- > The user can access the mirrored data points and process the values.

The following data is available:

- Parameters, status and diagnostic information of the gateway (e.g. ifm IO-Link master)
- Process data of the sensors / actuators connected to the ports of the gateway

Showing the ifm IoT Core tree of the edgeController

61723

To check which structures and data points are mirrored in the ifm IoT Core of the edgeController, the user can use the "IoTCore Visualizer" of the edgeController.

To show the "IoT Core Visualizer":

- ▶ Start web browser.
- ▶ Go to the following address: `http://<IP-address-edgeController>:8090/browse`
- > The browser shows the IoT Core Visualizer of the edgeController.

Offboarding a device

61724

To disable the mirroring of an ifm IoT Core enabled device, the user must offboard the device.

Available function blocks:

Name	Description
FB_offboard (FB) (→ p. 64)	The function block removes an ifm IoT Core enabled device from the ifm IoT Core of the edgeControllers (offboarding).

Requirement:

- The ifm IoT Core enabled device is connected to edgeController.
- ▶ Offboard the device
- > The mirroring of the device in the ifm IoT Core of the edgeController will be disabled.

- > Process data of the ifm IoT Core enabled device and the connected sensors / actuators can no longer be transferred to cloud systems.

7.4.5 Managing devices

61725

Various management functions can be performed on an onboarded unit.

Available function blocks:

Name	Description
FB_CheckOnlineState (FB) (→ p. 50)	The function block checks the online status of an IoT Core enabled device.
FB_GetEthernetAdapter (FB) (→ p. 56)	The function block provides the Ethernet adapters of a device.
FB_IoTDeviceList (FB) (→ p. 59)	The function block is used to manage a list of ifm IoT Core enabled devices.
FB_GetSensor (FB) (→ p. 58)	The function block reads the information of all sensors / actuators connected to an ifm IoT Core enabled device.
FB_MonitorSensor (FB) (→ p. 61)	The function block checks for all sensors / actuators connected to an ifm IoT Core enabled device whether the setpoint and actual values for Vendor ID (VID) and Device ID (DID) are matching.
FB_ExecuteScan (FB) (→ p. 53)	The function block searches the Ethernet network for available ifm IoT Core enabled devices.

7.4.6 Manual creation of data points in the ifm IoT Core

61726

The user has the option to create data points manually in the ifm IoT Core of the edgeController. The CODESYS application can then access these values (e.g. implementation of a counter).

Available function blocks:

Name	Description
FB_SetNode (FB) (→ p. 67)	The function block creates a structure in an ifm IoT Core of the edgeController and optionally writes a value.

- ▶ Declaring a variable in CODESYS.
- ▶ Assign a value to a variable (static or dynamic).
- ▶ Create a data point in the ifm IoT Core of the edgeController and link it to a CODESYS variable.



The declared variable and the created data point must be of the same variable type.

Deleting a data point in the ifm IoT Core

61727

The user can manually delete created data points in the ifm IoT Core of the edgeController.

Available function blocks:

Name	Description
FB_DeleteNode (FB) (→ p. 51)	The function block deletes a data point in the ifm IoT Core of the edgeController.

- ▶ Delete the required data point.
- > The values of the deleted data point will no longer be transferred to cloud systems.

7.4.7 Reading or writing data points of the ifm IoT Core

61728

The user has read or write access to data points of the ifm IoT Core of the edgeController.

Available function blocks:

Name	Description
FB_GetData (FB) (→ p. 55)	The function block cyclically reads the value of a data point in the ifm IoT Core of the edgeController and outputs it.
FB_SetData (FB) (→ p. 66)	The function block writes the value of a data point in the ifm IoT Core of the edgeController.

Requirements:

- The sensor / actuator is connected with the ifm IoT Core enabled device.
- The ifm IoT Core enabled device is connected with the edgeController.
- The ifm IoT Core enabled device has been onboarded.
- ▶ Read or write the value of the required data point (observe access rights).

7.4.8 Tagging data points for cloud systems

61729

For transfer to cloud systems, the individual data points must be tagged with a profile. Based on the assigned profile, the application recognises to which cloud system the data value is to be transferred. The following profiles are defined in the project template as an example. The pre-set profiles can be changed:

- c1oud1: Amazon AWS
- c1oud2: Mirasoft AnyViz
- c1oud3: Microsoft Azure

Available function blocks:

Name	Description
FB_TagProfiles (FB) (→ p. 69)	The function block assigns at least one profile to a data point.
FB_SetNode (FB) (→ p. 67)	The function block creates a structure in an IoT tree and optionally writes a value.

- ▶ Tag the data point with the required profile.
- > The data point is tagged for transfer to the corresponding cloud system.

7.4.9 Enabling the OPC-UA server

61730

The device offers the functions of an OPC UA server. The OPC UA server function is to be enabled and disabled separately.

Available function blocks:

Name	Description
FB_ActivateOPC (FB) (→ p. 49)	The function block enables the OPC UA servers of the ifm IoT Core of the edgeController.

- Enable the OPC-UA server.
- > OPC UA client can access the edgeController.

7.4.10 Transferring data to the cloud

61731

To transmit data to the cloud systems, the edgeController uses the MQTT protocol.

Transferring the data values of the IoT tree to a cloud system is a multi-step process.



The project template includes the program block "FB_SubscribeDataFromIoTCore (PRG)". The function block implements this multi-stage process in an exemplary manner.

The program block is commented.

- Familiarise yourself with the program block functions!
- Adjust FB_SubscribeDataFromIoTCore (PRG) to the requirements of the application.

Available function libraries:

Name	Description
FB_GetAddressesByProfile (FB) (→ p. 54)	The FB searches the ifm IoT Core of the edgeController for data points that are tagged with a specific profile and outputs the addresses of the data points in a list.
FB_EventHandling (FB) (→ p. 52)	The function block creates a user-defined event. The function block then subscribes to the event and determines which data will be sent when the event occurs.
FB_MQTT_Enable (FB) (→ p. 62)	The function block enables or disables the MQTT connection.
FB_MQTT_Subscribe (FB) (→ p. 63)	The function block subscribes to an MQTT topic and manages the notifications received.

Requirements:

- All ifm IoT Core enabled devices have been onboarded.
- All data to be transferred is tagged with the corresponding cloud profile.
- Collect all data points tagged with the cloud profile and save them in an array (FB_GetAddressesByProfile)
- Subscribe to data of the array (FB_EventHandling)
- Enable the MQTT connection and connect it to the MQTT broker (MQTTEnable)
- Transfer data of the array to the required cloud system if there is a new value (FB_GetJSONCloudData)
- > Data will be transferred cyclically to the cloud system.

7.5 Cloud systems

61732

7.5.1 Example application: clouds

61733

The project template includes a program blocks (PRG) for each supported cloud system. The program blocks establish the connection to the respective cloud system and control the data transfer. The program blocks of the individual cloud systems are called in the program block PRG_CloudConnect.



There are comments to the program blocks.

- Familiarise yourself with the program block functions!
- Adjust the program blocks to the requirements of the applications.

7.5.2 Function elements

61734

The **Library: ifm_iot.library** (→ p. 42) includes the following elements for connection to cloud systems:

Function element	Description
FB_CloudGetJSONVal (FB) (→ p. 78)	The function block reads a JSON key and outputs the value as a WSTRING.
PRG_CloudConnect (PRG) (→ p. 76)	The program block establishes the connection to the selected cloud systems.
CloudRemData (STRUCT) (→ p. 77)	Structure with non-volatile data of the supported cloud system
RemAnyViz (STRUCT) (→ p. 77)	Non-volatile data for cloud system Mirasoft AnyViz
RemAzure (STRUCT) (→ p. 77)	Non-volatile data for cloud system Microsoft Azure
RemAWS (STRUCT) (→ p. 77)	Non-volatile data for cloud system Amazon AWS
GVL_IoTCore (GVL) (→ p. 47)	Global variable list

The package also includes function libraries for communication with the supported cloud systems.

Library	Description
AnyViz Cloud Adapter	Functions for the connection to the Mirasoft AnyViz Cloud
AWS IoT Core Client SL	Functions and data structures for the connection to the Amazon AWS cloud
Azure IoT Hub Client SL	Functions and data structures for the connection to the Microsoft Azure cloud

The package also includes function libraries for communication via HTTP and MQTT.

Library	Description
HTTP Client SL	Functions and data structures for sending HTTP requests to an HTTP server
MQTT Client SL	Functions and data structures to connect the controller to an MQTT broker and send and receive messages



Documentation of the function libraries: Online help of the CODESYS Development System

7.5.3 Configuring the cloud adapter

61735

The AE1300 supports the following cloud computing platforms:

- Microsoft Azure (→ azure.microsoft.com)
- Amazon AWS (→ aws.amazon.com)
- Mirasoft AnyViz (→ www.anyviz.io)

Configuring the firewall

61736

- ▶ The following port numbers are enabled for data transfer to the cloud systems in the firewall:
 - MQTT: 8883
 - HTTPS: 443
 - HTTP: 8443

Setting up the AnyViz adapter

61737

For the edgeController, a logical device must be created in AnyViz.

The following data is required to set up the connection to the AnyViz cloud:

Requirements:

- The account on AnyViz portal has been set up and is enabled.
- The necessary port releases have been set up (→ **Configuring the firewall** (→ p. [27](#))).
- ▶ Note the following information in the AnyViz portal:
 - AnyViz project ID
 - AnyViz password
- ▶ In CODESYS: Create instance of the structure variable RemAnyViz (→ **RemAnyViz (STRUCT)** (→ p. [77](#))).
- ▶ Enter the following variable values in the instance:
 - AnyViz project ID
 - AnyViz password
 - Optional: Update interval
- ▶ In application: Program the transfer in the AnyViz cloud (→ **Transferring data to the cloud** (→ p. [25](#))).
- ▶ Transfer the application to the edgeController and start it.

The AnyViz portal automatically detects the edgeController when the application is running on the edgeController. The portal shows the status of the connection status. The edgeController is represented in the AnyViz portal via the MAC address.

Once the connection to the edgeController is established, the user can include data points from the edgeController on dashboards in the AnyViz portal.

Setting up the AWS adapter

Requirements:

- The account in the AWS portal has been set up and is enabled.
- An object of the type "IoT Hubs" has been created in the AWS portal.
- The CODESYS addon "Security Agent" has been installed (CODESYS Store).
- OpenSSL has been installed on CODESYS computers (<https://wiki.openssl.org/index.php/Binaries>).
- The necessary port releases have been set up (→ **Configuring the firewall** (→ p. [27](#))).
- ▶ Create a "Thing" object in the AWS portal.
 - Name: [edgeController] (example)
 - Type: "CODESYS" or "Things connected with CODESYS"
- ▶ Save Thing without a certificate.
- > The object has been created.
- ▶ The object "Thing" is open.
- ▶ Copy the HTTPS endpoint (e.g. "xxxxxxxxxxx-ats.iot.eu-central-1.amazonaws.com").
- ▶ In the AWS portal: Under [Secure] > [Policies], generate a rule (policy) with the following properties:
 - Name: ALL
 - Action: iot.*
 - Resource ARN: *
 - Effect: Allow
- ▶ In CODESYS: Create an instance of the RemAWS structure variable (→ **RemAWS (STRUCT)** (→ p. [77](#))).
- ▶ Enter the following variable values in the instance:
 - sClientId: Thing name (in the example: edgeController)
 - sHttpsHostname: HTTPS end point (in the example: xxxxxxxxxxxxxx-ats.)
 - Optional: sValueInterval: Update interval
- ▶ Transfer the application to the edgeController and start it.
- ▶ In the CODESYS project: Double-click on the [Device(edgeController)] node.
- ▶ Select [PLC Shell].
- ▶ Call up the command "cert-getapplist" on the device.
- > The component with the selected device name (in the example: edgeController) and a number will be displayed.
- ▶ Call up the command "cert-createcsr <Nummer>" on the device.
- > edgeController creates a CSR file in the directory "cert/export".
- ▶ Copy the generated CSR file to the local directory on the CODESYS computer (e.g. "C:/User/Desktop").
- ▶ On the CODESYS computer: Open the terminal window.
- ▶ Enter the following:


```
openssl.exe req -in " C:\User\Desktop _ edgeController.csr" -inform der -out C:\User\edgeController.csr"
```
- > OpenSSL generates a csr file.
- ▶ In the AWS portal: Open Thing and select the security panel.
- ▶ Select [View other options] > [create with csr].
- ▶ Select the csr file generated with OpenSSL.
- ▶ Click on [Upload].

- > The AWS portal provides a certificate (*.txt).
- ▶ Click on the link and download the root certificate. Select the following certificates:
 - Amazon root ca1
 - Amazon root CA3
 - Startfield Root CA
- ▶ Click on [Activate].
- ▶ Click [Attach a policy] and select the previously created policy.
- ▶ Repeat the following steps if necessary:
 - Select [Actions] > [Activate].
 - [Actions] > [Attach Thing] > [edgeController] > [Attach].
- ▶ Click on [Manage] > [Thing] > [edgeController] and check whether a certificate is displayed under [Security] and marked as active.
- ▶ In CODESYS: Select [View] > [Security screen].
- > The security settings appear.
- ▶ Select [Devices] > [Own certificates].
- ▶ Change the file extension of the downloaded certificate from *.txt to *.crt.
- ▶ Select the certificate (*.crt) and click on [Open].
- > CODESYS loads the certificate to the edgeController.
- ▶ Install all 3 certificates on the device.
- ▶ Reboot the edgeController.
- ▶ In CODESYS: Create an instance of the RemAWS structure variable (→ **RemAWS (STRUCT)** (→ p. [77](#))).
- ▶ Enter the following variable values in the instance:
 - AWS Client ID (device name)
 - AWS host name
 - Optional: Update interval
- ▶ In application: Programme the transfer in the AWS cloud (→ **Transferring data to the cloud** (→ p. [25](#))).
- ▶ Transfer the application to the edgeController and start it.

Setting up the Azure adapter

61739

Requirements:

- The account in the Azure portal has been set up and is enabled.
- An object of the type "IoT Hub" has been created in the Azure portal.
- Azure IoT Explorer has been installed (→ github.com/Azure/azure-iot-explorer/releases).
- The necessary port releases have been set up (→ **Configuring the firewall** (→ p. [27](#))).
- ▶ In the Azure portal: Add a new device to the "IoT Hub".
- ▶ Name new device (e.g. edgeController) and save it with default values.
- > Digital twin of the edgeController has been created.
- ▶ Select [Settings] > [Shared access policies].
- ▶ Under [Policy], select the value [iothubowner].
- ▶ Copy the [Connection string-primary key].
- ▶ Launch "Azure IoT Hub Explorer".
- ▶ Paste the copied value [Connection string-primary key].
- ▶ Click on [Connect].
- > The software establishes the connection to the Azure portal.
- ▶ Select the newly created device (in the example: edgeController).

- ▶ Select [Device identity].
- ▶ Generate SAS token.
- ▶ Copy the area starting from "SharedAccessSignature" of the SAS token. Example (section marked in bold):

HostName=ifmRSTIoTHub.azure-devices.net;DeviceId=TestDevice;SharedAccessSignature=**SharedAccessSignature**
sr=ifmRSTIoTHub.azure-devices.net%2Fdevices%2FTestDevice&sig=%2FSDiCS%2Bhv89c8Zk
%2BKvLclZLhHfv2SVcUh9DxCLSDB2o%3D&se=1593387430

- ▶ Create the text file "AzureSAS.txt" and paste the copied SAS token. The file must not contain any line breaks.
- ▶ Copy the text file to the root directory of a USB memory device.
- ▶ Connect the USB memory with the edgeController.
- ▶ In the CODESYS project: Create the instance of the structure variable RemAzure (→ **RemAzure (STRUCT)** (→ p. [77](#)))
- ▶ Enter the following variable values in the instance:
 - Name of the IoT Hub
 - Name of the device
 - SAS token
 - Optional: Update interval
- ▶ In application: Program the transfer to the Azure cloud (→ **Transferring data to the cloud** (→ p. [25](#))).
- ▶ Transfer the application to the edgeController and start it.
- > The Azure adapter is set up.
- > The edgeController can send data to the Azure cloud.

7.5.4 Establishing the connection to the cloud systems

61740

Requirements:

- The cloud adapters have been correctly configured (→ **Configuring the cloud adapter** (→ p. [27](#))).
- ▶ Adjust the program block "PRG_CloudConnect (PRG)" to the requirements of the application.
- ▶ Integrate the program block "PRG_CloudConnect (PRG)" into the application.

7.6 Visualisation

61741

Visualisations allow the programmer to create a graphical user interface for the application.



Become familiar with concepts and basics of visualisation in CODESYS (→ CODESYS online help)!

7.6.1 Creating a new visualisation page

61742

To create a visualisation for an application:

Requirements:

- The application is open.
- ▶ In the device tree: Select the node [Application].
- ▶ Select [Project] > [Add Object] > [Visualisation...].
- > A dialogue window appears.
- ▶ Enter a unique name for the new visualisation (e.g. MyVisu).
- > The new visualisation appears in the device tree.
- ▶ Double-click [MyVisu].
- > The visualisation editor with the window [Tools] appears.
- ▶ Create the required visualisation.
- ▶ When creating the visualisation, follow the instructions in the following sections!
- ▶ Save the project to adopt the changes.

7.6.2 Supported visualisation types

61743

The device supports the following visualisation types:

- Target-Visu: Visualisation on the device display
- Web-Visu: Visualisation in the web browser

7.6.3 Visualisation tasks

61744

Basic tasks of a visualisation can be:

- visualising process data
 - displaying current process values
 - displaying process value curves
- visualising diagnostic and status information
 - showing the status of the cloud connection
 - displaying the data transfer to cloud systems
- changing the control values of a process
- starting functions
 - starting a network scan
 - starting the onboarding / offboarding of IoT Core enabled devices

7.6.4 Use image collection

40643



- ▶ Familiarise yourself with the following CODESYS functions!
 - Image collection
 - Online help > CODESYS Development System > Program application > Use image collection

To use own images in visualisations the user must first add the corresponding image files to an image collection in the application.

To add an image collection to a project:

- ▶ In the device tree: Mark node [Application].
- ▶ Select [Project] > [Add object] > [Image Pool...].
- > Window [Add Image Pool] appears.
- ▶ Enter the name of the image collection and confirm with [Add].
- > Image collection appears in the project tree as a subelement of the application.

Image properties

57751



The following image file formats are supported:

- BMP
- JPG
- PNG
- SVG
- TIF

7.6.5 Example application: visualisation

61745

The project template includes the visualisation of the start application as an example.

- ▶ In the project tree: double-click on the node [Application] > [Visu_Eth_Settings].
- > The visualisation of the startup application appears.

7.6.6 Visualising cloud operations

61746

The function library "ifm_iiot.library" provides the following function elements for the visualisation of various cloud functions.

Function element	Description
FB_DataPointsTable (FB) (→ p. 71)	The function block is necessary for the presentation and logic of the visualisation side of the data points of the individual cloud systems.
FB_DeviceTable (FB) (→ p. 72)	The function block supports the visualisation and logic of the device table and the device configuration.
FB_flyingNumber (FB) (→ p. 74)	The function block visualises the data transmission (flying numbers).
FB_Move (FB) (→ p. 75)	The function block controls the movement of elements in the visualisation.
POC (STRUCT) (→ p. 70)	The structure provides information about the touch point on the display.

7.6.7 Setting the properties of the visualisation

61747

In the [Visualisation Manager], the user can set the properties of the visualisations.

To set the properties of the target visualisation:

Requirements:

- The visualisation has been created.
- ▶ In the project: Double-click [Application] > [Visualisation Manager] > [TargetVisu].
- > The editor for properties of the target visualisation appears.
- ▶ Select the newly created visualisation in the [Start visualisation] field.
- ▶ When using the display in portrait format: Adjusting the size of the visualisation
 - Client width: 480
 - Client height: 1280
- ▶ Do not change any other settings.
- ▶ Save the project.

7.7 Task configuration

61748

The processing of the application components is controlled via tasks. The properties of a task can be set via parameters.

The project template includes pre-configured tasks for the different components of the application.

Task	Description
Main task (IEC tasks)	Task for the processing of the main program [PLC_PRG (PRG)]
Task_CloudConnect (IEC-Tasks)	Task for the processing of the cloud communication
Task_IoTCore (IEC tasks)	Task for processing the IoT core functions
VISU_Task (IEC tasks)	Task for the processing of the visualisation functions



Familiarise yourself with the CODESYS task configuration (→ CODESYS online help)!

7.7.1 Setting the task configuration

61749

ifm recommends using the task structure of the project template.

- ▶ Assign the newly created program blocks and visualisation pages to the predefined categories.
- ▶ Do not change the settings of the tasks!

8 Operation

Content

Transfer CODESYS project to the device	35
Operating states of the PLC application	37
Reset	38
Display system information.....	40

40465

8.1 Transfer CODESYS project to the device

Content

Load application to AE1300	35
Delete application from the device	36

40608



- ▶ Familiarise yourself with the following CODESYS functions!
 - Transfer application to the device
→ Online help > CODESYS Development System > Transfer application to the PLC
 - Monitoring
→ Online help > CODESYS Development System > Application during the runtime > Monitoring of values
- ▶ Observe the notes about the states of the PLC application!
→ **Operating states of the PLC application** (→ p. [37](#))

8.1.1 Load application to AE1300

40481

To load the created application to the device and store it non-volatily:

Prerequisites:

- > Connection between PC/laptop and AE1300 established.
- > Communication path has been set (→ **Set communication path of PLC**).
- > Project has been tested.

1 Translate application

- ▶ In the project tree: Click on [Application].
- ▶ Select [Build] > [Build].
- > CODESYS creates program code of the application.

2 Upload application to the device

- ▶ Select [Online] > [Login].
- > CODESYS changes to the online mode.
- > CODESYS loads active application to the device (download).
- > Application on the device is in the STOP state.

3 Start application

- ▶ Select [Debug] > [Start].
- > Application on the device changes to the RUN state.

4 Create boot application

- ▶ Select [Online] > [Create boot application].
- > CODESYS stores the application non-volatily on the device.

8.1.2 Delete application from the device

40916

To delete an application stored on the device:

1 Connect with the device

- ▶ In the device tree: Click on [Application].
- ▶ Select [Online] > [Login].
- > CODESYS changes to the online mode.

2 Delete application

- ▶ In the device tree: Double-click on [Device (AE1300)]
- > Editor window shows device settings.
- ▶ Select tab [Applications].
- ▶ Click on [Liste aktualisieren].
- > Editor window shows the applications stored on the device.
- ▶ Click on [Remove All] to delete all applications.
OR:
Select requested application and click on [Remove] to delete individual applications.
- > CODESYS deletes the selected applications from the device.

8.2 Operating states of the PLC application

Content

Display operating mode of the PLC application	37
Start PLC application	37
Stop PLC application	37

40466

The applications stored on the AE1300 are executed independently from each other in separate tasks. An application can have the following operating states:

- **Unload**
No application is stored on the device.
- **RUN**
 - The application is stored on the device.
 - The application is processed cyclically.
- **STOP**
 - The application is stored on the device.
 - The application is not processed.

8.2.1 Display operating mode of the PLC application

40494

To display the current operating status of the PLC application stored on the device:

> In the device tree: Symbol [Application] shows the current status

OR:

> In the online mode: CODESYS status bar shows the current state of the application.

8.2.2 Start PLC application

40623

To start the execution of the PLC application:

- ▶ In the device tree: Right-click on [Application] and select [Aktive Applikation setzen].
- ▶ Select [Online] > [Login].
- > CODESYS changes to the online mode.
- ▶ Select [Debug] > [Starten].
- > Application changes to the RUN state (→ **Operating states of the PLC application** (→ p. [37](#))).
- ▶ Optional: Repeat process for further applications.

8.2.3 Stop PLC application

40630

To stop the execution of the PLC application:

- ▶ In the device tree: Right-click on [Application] and select [Aktive Applikation setzen].
- ▶ Select [Online] > [Login].
- > CODESYS changes to the online mode.
- ▶ Select [Debug] > [Stop].
- > Application changes to the STOP state (→ **Operating states of the PLC application** (→ p. [37](#))).
- ▶ Optional: Repeat process for further applications.

8.3 Reset

Content

Supported reset variants	38
Reset application (warm)	38
Reset application (cold)	39
Reset application (origin)	39

39674

8.3.1 Supported reset variants

39592

The following table shows the reset variants supported by the device-internal CODESYS PLC and the resulting system behaviour:

Type of reset	System behaviour	Triggering actions
Reset (warm)	- application goes to STOP state. - Standard variables (VAR) of the application are initialised. - Remanent variables (VAR RETAIN) of the application keep their current values.	→ Reset application (warm) (→ p. 38)
Reset (cold)	- application changes to the STOP state. - All variables (VAR, VAR RETAIN) of the application are initialised.	→ Reset application (cold) (→ p. 39)
Reset (default)	- application goes to STOP state. - The application on the PLC is deleted. - All variables (VAR, VAR RETAIN) of the application are initialised. - PLC is reset to the default state.	→ Reset application (origin) (→ p. 39)



A variable that has been declared without an initialisation value is initialised with the variable-specific standard value (e.g. INT = 0).

8.3.2 Reset application (warm)

39671

To reset the application:

- ▶ In the device tree: Select [Application] and select
- ▶ [Online] > [Login] as active application.
- > CODESYS changes to the online mode.
- ▶ Select [Online] > [Reset warm] to reset the application.
- > Application changes to the STOP state.
- > Standard variables are newly initialised.
- > Retain variables keep their values.

8.3.3 Reset application (cold)

39675

To reset the application:

- ▶ In the device tree: Select [Application].
- ▶ Select [Online] > [Login].
- > CODESYS changes to the online mode.
- ▶ Select [Online] > [Reset cold] to reset the application.
- > Application changes to the STOP state.
- > All variables are newly initialised

8.3.4 Reset application (origin)

39670

To reset the application:

- ▶ In the device tree: Select [Application].
- ▶ Select [Online] > [Login].
- > CODESYS changes to the online mode.
- ▶ Select [Online] > [Reset origin] to reset the application.
- > Application changes to the STOP state and is deleted.
- > All variables are newly initialised
- > PLC is reset to the original state.

8.4 Display system information

In the online mode the device tree displays the current values of the following system parameters:

Parameter	Description	Possible values
[IP Settings]	IP settings	--
▪ [IP Address]	IP address of the device	E.g. 192.168.0.100
▪ [IP Mask]	Subnet mask of the network	E.g. 255.255.255.0
▪ [Gateway Address]	IP address of the network gateway	E.g. 192.168.0.2
[Version Firmware]	Version of the installed firmware	E.g. V1.4.0
[Serial Number Device]	Serial number of the device	E.g. 1511AB019

To display the system information of the device:

- ▶ Establish connection between CODESYS and AE1300.
- ▶ Select [Online] > [Login].
- > CODESYS changes to the online mode.
- ▶ In the device tree: Double-click on [System_Info]
- ▶ In the editor window: Select tab [Parameter].
- > In the editor window: Table shows current values of the system parameters.

9

Appendix

Content	
Library: ifm_iot.library	42
Library ifmDevice_ecomatDisplay.library	79

33879

9.1 Library: ifm_iot.library

Content	
IoT Core.....	43
HMI_CloudConnect	70
Cloud	76

61751

9.1.1 IoT Core

Content	
STRUCT	44
ENUM	47
GVL_IoTCore (GVL)	47
FB_ActivateOPC (FB)	49
FB_CheckOnlineState (FB)	50
FB_DeleteNode (FB)	51
FB_EventHandling (FB)	52
FB_ExecuteScan (FB)	53
FB_GetAddressesByProfile (FB)	54
FB_GetData (FB)	55
FB_GetEthernetAdapter (FB)	56
FB_GetJSONCloudData (FB)	57
FB_GetSensor (FB)	58
FB_IoTDeviceList (FB)	59
FB_MonitorSensor (FB)	61
FB_MQTT_Enable (FB)	62
FB_MQTT_Subscribe (FB)	63
FB offboard (FB)	64
FB_onboard (FB)	65
FB_SetData (FB)	66
FB_SetNode (FB)	67
FB_TagProfiles (FB)	69

STRUCT

Content

CloudDatapoint (STRUCT)	44
IoTCloudConnector (STRUCT)	44
IoTDevice (STRUCT)	45
IoTRemData (STRUCT)	45
IoTSensor (STRUCT)	45
REQ (STRUCT)	46

61754

CloudDatapoint (STRUCT)

61755

Name	Data type	Description	Possible values
wsAddress	WSTRING (1000)	address of the data point	e. g. 'processdatamaster/temperature'
wsType	WSTRING (255)	data type of the data point	e. g. 'processdata'
wsValue	WSTRING (255)	Value of the data point	e. g. '34'

IoTCloudConnector (STRUCT)

61756

Name	Data type	Description	Possible values
▪ sNameCloud	STRING (20)	Cloud name	e. g. 'MS Azure'
▪ sProfile	STRING (10)	Cloud profile	e. g. 'cloud1'
▪ tSendTime	TIME	Update interval (value in ms)	e. g. 500

IoTDevice (STRUCT)

61757

Name	Data type	Description	Possible values
▪ sDeviceName	STRING (20)	Device name	e. g. 'AL1320'
▪ sMacAddress	STRING (20)	MAC address of the device	e. g. '00:02:01:05:50:8B'
▪ sIpAddress	STRING (20)	IP address of the device	e. g. '192.168.83.10'
▪ sSubnetMask	STRING (20)	Subnet mask of the device	e. g. '255.255.255.0'
▪ sDeviceID	STRING(5)	IO-Link device ID	e. g. '000469'
▪ sVendorID	STRING(5)	IO-Link vendor ID	e. g. '0136' (= ifm electronic gmbh)
▪ xBoardError	BOOL	Boarding error	- FALSE: no error - TRUE: error
▪ xManualDevice	BOOL	Manually added device	- FALSE: automatically detected - TRUE: manually added
▪ eBoardStatus	IoTDevice BoardStatus	boarding status of the device	IoTDeviceBoardStatus (ENUM) (→ p. 47)
▪ eConStatus	IoTDevice ConStatus	connection status of the device	IoTDeviceConStatus (ENUM) (→ p. 47)
▪ arrSensorsGroup	ARRAY OF IoTSensor	list of devices connected to IoT core	per line: IoTSensor (STRUCT) (→ p. 45)

IoTRemData (STRUCT)

61758

Name	Data type	Description	Possible values
▪ arDevList	ARRAY OF IoTDevice	list with detected and manually added devices	per line: IoTDevice (STRUCT) (→ p. 45)
▪ uiDevAct	UINT	current index of the list	e. g. '1'
▪ sUsedEthAdapter	STRING(15)	IP address of the Ethernet adapter used	e. g. '192.168.82.10'
▪ xOPCUAAActiv	BOOL	activate / deactivate OPC UA support of the IoT core	- FALSE: deactivated - TRUE: activated

IoTSensor (STRUCT)

61759

Name	Data type	Description	Possible values
▪ sProductName	STRING(20)	Sensor name	
▪ sDeviceID	STRING(5)	IO-Link device ID	e. e. '000469' (= TCC501)
▪ sVendorID	STRING(5)	IO-Link vendor ID	e. g. '0136' (= ifm electronic gmbh)
▪ sKonfigDeviceID	STRING(5)	Target device ID	0: no verification
▪ sKonfigVendorID	STRING(5)	Target vendor ID	0: no verification
▪ xErrorMonitor	BOOL	configuration error (wrong IDs)	- FALSE: no error - TRUE: error

REQ (STRUCT)

61760

Name	Data type	Description	Initial value
NO_REQ	INT		0
OWN_REQ	INT		1
GETIDENTITY	INT		1000
GETTREE	INT		1001
GETFILTEREDPROFILE	INT		1002
GETETHADAPTER	INT		1003
PERFORMSCAN	INT		1004
ABORTSCAN	INT		1005
MIRROR	INT		1006
UNMIRROR	INT		1007
EXTENDEDIODD	INT		1008
SETPROFILE	INT		1009
GETDATA	INT		1010
SETDATA_INT	INT		1011
SETDATA_REAL	INT		1012
SETDATA_STRING	INT		1013
TRIGGER_EVENT	INT		1014
SETNEWDATA_INT	INT		1015
SETNEWDATA_REAL	INT		1016
SETNEWDATA_STRING	INT		1017
ADDNODE	INT		1050
ADDNODE_DATA_INT	INT		1051
ADDNODE_DATA_REAL	INT		1052
ADDNODE_DATA_STRING	INT		1053
REMOVENODE	INT		1054
ADDNETADAPTER	INT		1055
SUBSCRIBE	INT		2000
UNSUBSCRIBE	INT		2001

ENUM

Content

IoTDeviceBoardStatus (ENUM).....	47
IoTDeviceConStatus (ENUM).....	47

IoTDeviceBoardStatus (ENUM)

61761

Name	Data type	Description	Possible values
IoTDeviceBoardStatus	INT	Boarding status	0: offboarded 1: onboarded

IoTDeviceConStatus (ENUM)

61762

Name	Data type	Description	Possible values
IoTDeviceConStatus	INT	Connection status	0: offline 1: online

GVL_IoTCore (GVL)

61764

Variable	Data type	Description	Initial value
arCloudProfiles	ARRAY[1..3] OF IoTCloudConnector	Array with properties of the supported cloud systems per line: → IoTCloudConnector (STRUCT) (→ p. 44)	
ui_AnzDevOnline	UINT	number of online devices	
ui_AnzDevOffline	UINT	number of offline device	
slpDisplay	STRING(15)	IP address of the edgeController	'192.168.82.121'
sPortIoTCore	STRING(4)	Port number of the IoT core	'8090'
arr_slpAdapter	ARRAY OF STRING(15)	Arrays with IP addresses of the Ethernet adapters	['10.1.1.100', '192.168.82.121']
bSelectedEthernetAdapter	BYTE	Number of the selected Ethernet adapter	1
eConStatelotCore	→ IoTDeviceConStatus (ENUM) (→ p. 47)	Connection status of the IoT Core of the edgeController	
arAddressListTagClouds	ARRAY[1..3,1..100] OF STRING(255)	two-dimensional array; contains a list of addresses of the data points to be transferred for each cloud system	
xMQTTConnectedToBroker	BOOL	connection status to the MQTT broker	
arCloudData	ARRAY[1..3] OF STRING(4096)	array with cloud data of the IoT Core as STRING/JSON	
arwsCloudData	ARRAY[1..3] OF WSTRING(4096)	array with cloud data of the IoT cores as WSTRING/JSON	
arCloudDataReady	ARRAY[1...3] OF BOOL	array with signals for provision of new data	

Variable	Data type	Description	Initial value
arwsCloudPayload	ARRAY[1..3,1..110] OF CloudDatapoint	two-dimensional array; contains the read JSON data for each supported cloud system per JSON date: → CloudDatapoint (STRUCT) (→ p. 44)	
aruiCountDatapoints	ARRAY[1..3] OF UINT	array with number of cloud data points	
xIoTState	BOOL	overall connection status (all devices and edgeController)	▪ 0: error ▪ 1: no error
uiMaxAnzDevices	UINT	max. number of devices	40
uiMaxAnzIpAdapters	UINT	max. number of IP adapters	5
uiMaxAnzIoTSENSOREN	UINT	max. number of IoT sensors	8
uiMaxAnzCloudProfiles	UINT	max. number of cloud profiles	3
uiMaxAnzAdresses	UINT	max. number of addresses	110
uiMaxAnzTagAdresses	UINT	max. number of tag addresses	100
uiMaxStringSize	UINT	max. number of characters of a STRING	4096

FB_ActivateOPC (FB)

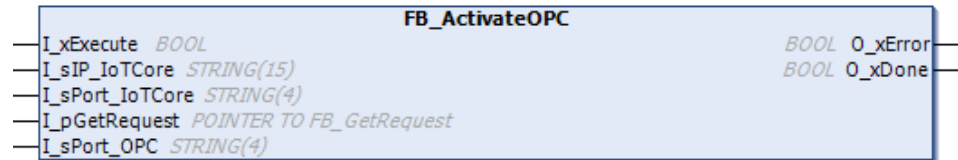
61766

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61768

The function block enables the OPC-UA servers of the ifm IoT core of the edgeController.

Inputs

61769

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_sPort_OPC	STRING(4)	Port number of the OPC server	

Outputs

61771

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed

FB_CheckOnlineState (FB)

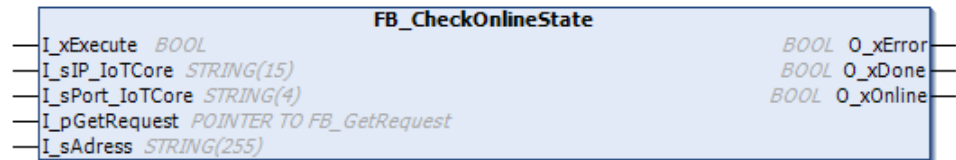
61774

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61775

The function block checks the online status of an ifm IoT Core enabled device by querying the identity of the device with the getidentity service.

Inputs

61776

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_sAddress	STRING(255)	Device address	

Outputs

61777

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed
O_xOnline	BOOL	online status	- FALSE: Not Online - TRUE: online

FB_DeleteNode (FB)

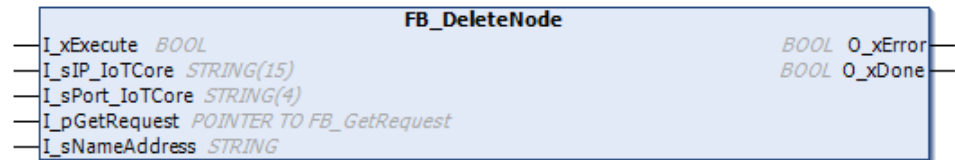
61779

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61780

The function block deletes a data point in the ifm IoT Core of the edgeController.

Inputs

61781

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_sNameAddress	STRING(255)	address of the data point	

Outputs

61771

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed

FB_EventHandling (FB)

61783

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61784

The function block creates a user-defined event. The function block then subscribes to the event and determines which data will be sent when the event occurs. In addition, the update rate will be set.

Inputs

61785

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_tSentTime	TIME	transmission interval (value in ms)	e.g. 1000
I_sNameStructure	STRING(255)	name of the structure, subnodes separated by /	
I_parNameEventDatapoint	POINTER	array with data points whose values are to be transmitted (DataToSend)	
I_xIncludeStructOnlyOnce	BOOL	create structure node once	- FALSE: no transmission - TRUE: transmission
I_sSubscribeTopic	STRING(255)	MQTT topic that triggers a notification	
I_xResetEvent	BOOL	unsubscribe from notifications	- FALSE: do not reset - TRUE: reset

Outputs

61786

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed
O_xBusy	BOOL	activity display of the FB	- FALSE: function block is inactive - TRUE: function block is active

FB_ExecuteScan (FB)

61788

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61789

The function block searches the Ethernet network for available ifm IoT Core enabled devices.

Inputs

61790

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_sEthAdapter	STRING(15)	name of the Ethernet interface	- ETH0: interface ETH0 - ETH1: interface ETH1
I_sNameScanner	STRING(25)	scanner name	e. g. eip_scanner or dcp_scanner
I_sMQTTTopicName	STRING	MQTT topic name	e. g. temperature

Outputs

61771

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed

FB_GetAddressesByProfile (FB)

61792

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:

```

FB_GetAddressesByProfile
- I_xExecute: BOOL
- I_sIP_IoTCore: STRING(15)
- I_sPort_IoTCore: STRING(4)
- I_pGetRequest: POINTER TO FB_GetRequest
- I_sProfile: STRING
- I_pJSON_FindKey_O: POINTER TO FB_JSON_FindKey
- O_xError: BOOL
- O_xDone: BOOL
- O_arwsAdrlst: ARRAY[1..GVL_IoTCore.uiMaxAnzAddresses] OF WSTRING(255)

```

Description

61793

The FB searches the ifm IoT Core of the edgeController for data points that are tagged with a specific profile and outputs the addresses of the data points in a list.

Inputs

61794

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_sProfile	STRING	Profile	e. g. 'cloud1'
I_pJSON_FindKey_O	POINTER	pointer to FB_JSON_FindKey	

Outputs

61795

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed
O_arwsAdrlst	ARRAY of WSTRING	list with addresses of the data points	

FB_GetData (FB)

61797

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61798

The function block cyclically reads the value of a data point in the ifm IoT Core of the edgeController and outputs it.

Inputs

61799

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_sAddress	STRING(255)	address of the data point	e.g. 'devicetag/applicationtag'
I_tPollTime	TIME	query time	e. g. 500
I_bSentDataType	BYTE	data type of the data point	0: INT 1: REAL 2: STRING
I_pJSON_FindKey_O	POINTER	pointer to FB_FindKey	

Outputs

61800

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed
I_iValueDatapoint	INT	INT value of the data point	e.g. 23
I_rValueDatapoint	REAL	REAL value of the data point	e.g. 2.34
I_sValueDatapoint	STRING	STRING value of the data point	e.g. 'AE1300'

FB_GetEthernetAdapter (FB)

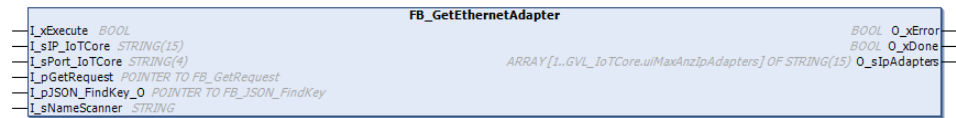
61802

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61803

The function block provides the Ethernet adapters of a device.

Inputs

61804

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_pJSON_FindKey_O	POINTER	pointer to FB_JSON_FindKey	
I_sNameScanner	STRING	scanner name	e.g. eip_scanner

Outputs

61805

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed
O_slpAdapters	ARRAY OF STRING(15)	list of Ethernet adapters of the device	e.g. ["ETH0", "ETH1"]

FB_GetJSONCloudData (FB)

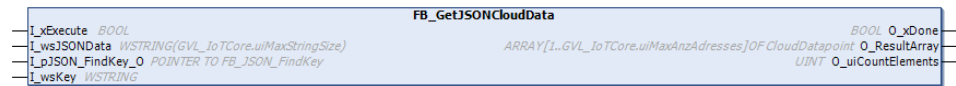
61807

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61808

The function block extracts a specific data point from cloud data in JSON format; the function block is required for the display in the visualisation.

Inputs

61809

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_wsJSONData	WSTRING	JSON cloud data of an event	
I_pJSON_FindKey_O	POINTER	pointer to FB_JSON_FindKey	
I_wsKey	WSTRING	search term	

Outputs

61810

Output	Data type	Description	Possible values
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed
O_ResultArray	ARRAY	array with data points	
O_uiCountElements	UINT	number of elements in the array	

FB_GetSensor (FB)

61812

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61813

The function block reads the information of all sensors / actuators connected to an ifm IoT Core enabled device. The results will be saved on a data structure.

Inputs

61814

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_pJSON_FindKey_O	POINTER	pointer to FB_JSON_FindKey	
IO_Device	POINTER	pointer to data structure with information about the IoT Core enabled device and the connected sensors / actuators	IoTDevice (STRUCT) (→ p. 45)

Outputs

61771

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed

FB_IoTDeviceList (FB)

61816

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61817

The function block is used to manage a list of ifm IoT Core enabled devices. The following operations can be performed on the list:

- manually adding an ifm IoT Core enabled device
- setting an ifm IoT Core enabled device offline
- deleting an offboard device from the list
- deleting all devices from the list
- setting the connection status of all devices to "offline"
- updating the connection status of the devices

Inputs

61818

Input	Data type	Description	Possible values
I_xAddManual	BOOL	manually adding an ifm IoT Core enabled device	- FALSE: do not add - TRUE: add manually
I_xAddIoTDeviceEvent	BOOL	add an ifm IoT Core enabled device to an event	- FALSE: do not add - TRUE: add
I_xSetIoTDeviceOffline	BOOL	set all ifm IoT Core enabled devices offline	- FALSE: do not set offline - TRUE: set offline
I_xDelete Offboarded	BOOL	delete all offboard devices from the list	- FALSE: do not delete - TRUE: delete
I_xUpdateConState	BOOL	update the connection status of the ifm IoT Core enabled devices	- FALSE: do not update - TRUE: update
I_xDeleteAll	BOOL	delete all ifm IoT Core enabled devices from the list	- FALSE: do not delete - TRUE: delete
I_IoTDevice	IoTDevice	ifm IoT Core enabled device	→ IoTDevice (STRUCT) (→ p. 45)
I_pIoTDevList	POINTER to ARRAY of IoTDevice	pointer to array with ifm IoT Core enabled devices	per line: → IoTDevice (STRUCT) (→ p. 45)
I_pIoTDevAct	POINTER to UINT	index of the ifm IoT Core enabled device	
I_pAnzIoTDevOnline	POINTER to UINT	number of online devices	
I_pAnzIoTDevOffline	POINTER to UINT	number of offline device	

Outputs

61819

Output	Data type	Description	Possible values
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed

FB_MonitorSensor (FB)

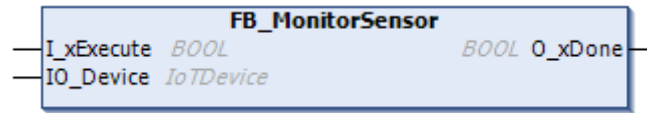
61821

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61822

The function block checks for all sensors / actuators connected to an ifm IoT Core enabled device whether the target and actual values for Vendor ID (VID) and Device ID (DID) are matching.

Inputs

61823

Designation	Data type	Meaning	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
IO_Device	POINTER to IoTDevice	pointer to data structure with information about ifm IoT enabled device (IO-Link master, sensor, actuator)	→ IoTDevice (STRUCT) (→ p. 45)

Outputs

61824

Designation	Data type	Meaning	Possible values
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed

FB_MQTT_Enable (FB)

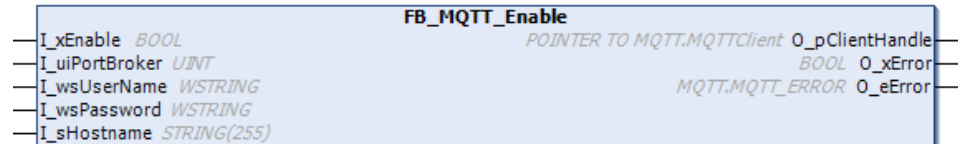
61826

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61827

The function block enables or disables the MQTT connection.

Inputs

61828

Input	Data type	Description	Possible values
I_xEnable	BOOL	FB control	- FALSE: Deactivate FB - TRUE: Enable FB
I_uiPortBroker	UINT	Port number of the MQTT broker	e.g. 1880
I_wsUserName	WSTRING	User name (optional)	e.g.. 'user'
I_wsPassword	WSTRING	Password (optional)	e.g. 'passwd'
I_sHostname	STRING	Host name	e.g.. 'localhost'

Outputs

61829

Designation	Data type	Meaning	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_eError	BOOL	MQTT error display	- FALSE: no error - TRUE: error
O_pClientHandle	POINTER	Pointer to MQTT client	

FB_MQTT_Subscribe (FB)

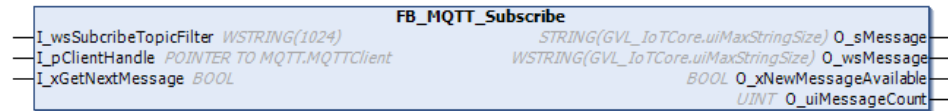
61831

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61832

The function block subscribes to an MQTT topic and manages the notifications received.

Inputs

61833

Input	Data type	Description	Possible values
<code>I_wsSubscribeTopicFilter</code>	WSTRING (1024)	Topic to subscribe	e.g. "processdatamaster/temperature"
<code>I_pClientHandle</code>	POINTER	Pointer to MQTT client	
<code>I_xGetNextMessage</code>	BOOL	Output the next notification in the buffer	- FALSE: current notification - TRUE: next notification

Outputs

61834

Output	Data type	Description	Possible values
<code>O_sMessage</code>	STRING	MQTT notification	
<code>O_wsMessage</code>	WSTRING	MQTT notification	
<code>O_xNewMessageAvailable</code>	BOOL	Display of new notifications	- FALSE: not available - TRUE: available:
<code>O_uiMessageCount</code>	UINT	Notifications counter	0...65535

FB offboard (FB)

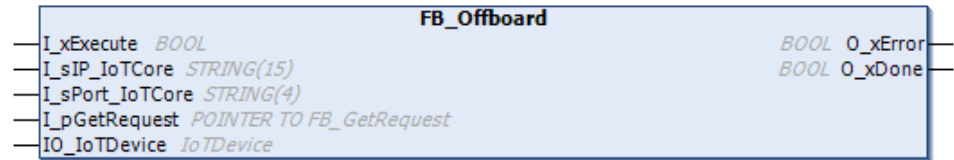
61836

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61837

The function block removes an IoT Core enabled device from the IoT Core of the edgeController (offboarding).

Inputs

61838

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
IO_IoTDevice	POINTER to IoTDevice	pointer to data structure with information about IoT Core enabled device	IoTDevice (STRUCT) (→ p. 45)

Outputs

61771

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed

FB_onboard (FB)

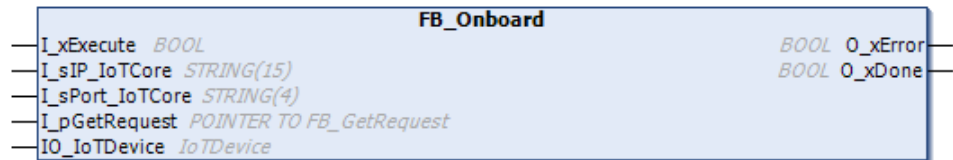
61840

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61841

The FB adds an IoT Core enabled device to the IoT Core of the edgeController (onboarding).

Inputs

61838

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
IO_IoTDevice	POINTER to IoTDevice	pointer to data structure with information about IoT Core enabled device	IoTDevice (STRUCT) (→ p. 45)

Outputs

61771

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed

FB_SetData (FB)

61843

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61844

The function block writes the value of a data point in the ifm IoT Core of the edgeController.

Inputs

61845

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_sAddressSetData	STRING(255)	address of the data point	e.g. 'devicetag/applicationtag'
I_bSentDataType	BYTE	data type of the data point	0: INT 1: REAL 2: STRING
I_iValueDatapoint	INT	INT value of the data point	e.g. 23
I_rValueDatapoint	REAL	REAL value of the data point	e.g. 2.34
I_sValueDatapoint	STRING	STRING value of the data point	e.g. "AL1350"

Outputs

61786

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed
O_xBusy	BOOL	activity display of the FB	- FALSE: function block is inactive - TRUE: function block is active

FB_SetNode (FB)

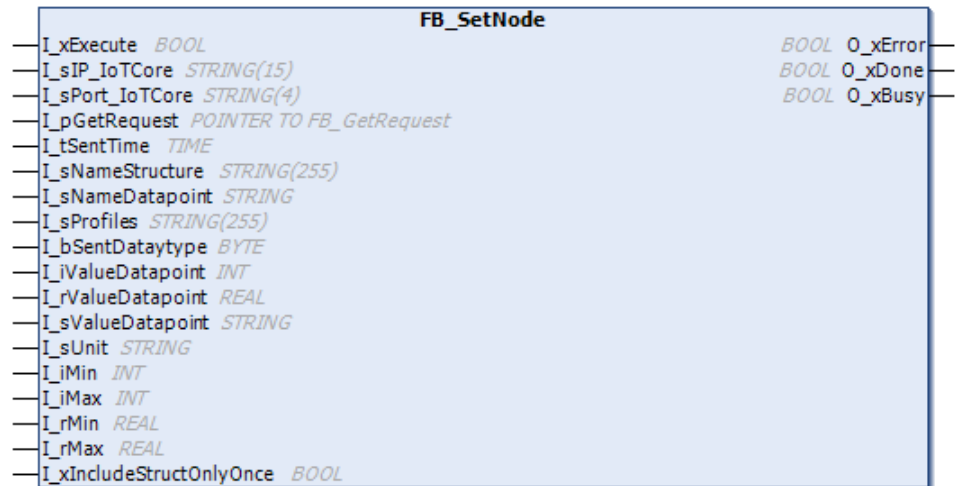
61847

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61848

The function block creates a structure in an ifm IoT Core of the edgeController and optionally writes a value.

Inputs

61849

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_tSentTime	TIME	interval time for cyclic transmission	
I_sNameStructure	STRING(255)	name of the structure in which the data point is to be created (separated by /)	e.g. 'codesys/mydata'
I_sNameDatapoint	STRING	name of the data point	e.g. 'mydatapoint'
I_sProfiles	STRING(255)	profile; several profiles separated by comma (optional)	e.g. 'cloud1'
I_bSentDatatype	BYTE	data type of the data point	0: INT 1: REAL 2: STRING
I_iValueDatapoint	INT	INT value	e.g. 3
I_rValueDatapoint	REAL	REAL value	e.g. 2.1
I_sValueDatapoint	STRING	STRING value	e.g. 'Counter'

Input	Data type	Description	Possible values
I_sUnit	STRING	Unit	e.g. '°C'
I_iMin	INT	Minimum INT value	e.g. 0
I_iMax	INT	Maximum INT value	e.g. 6
I_rMin	REAL	Minimum REAL value	e.g. 1.5
I_rMax	REAL	Maximum REAL value	e.g. 2.5
I_xIncludeStructOnlyOnce	BOOL	Create the structure only once	- FALSE: single - TRUE: recurring

Outputs

61786

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed
O_xBusy	BOOL	activity display of the FB	- FALSE: function block is inactive - TRUE: function block is active

FB_TagProfiles (FB)

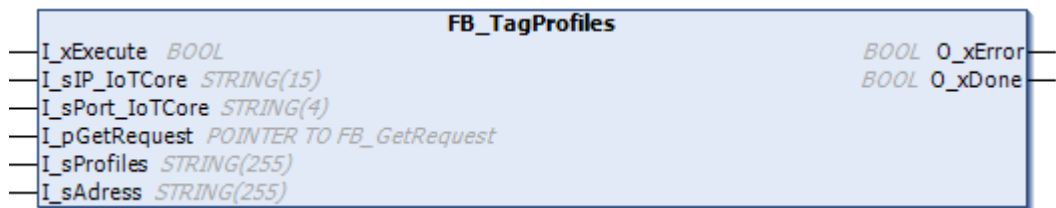
61851

Function block type: Function block (FB)

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61852

The function block assigns at least one profile to a data point.

Inputs

61853

Input	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_sIP_IoTCore	STRING(15)	IP address of the ifm IoT Core	e.g. '192.168.82.121'
I_sPort_IoTCore	STRING(4)	port number of the ifm IoT core	e.g. '80'
I_pGetRequest	POINTER	pointer to FB_GetRequest	
I_sProfiles	STRING(255)	profile; separate several profiles with a comma	e. g.: cloud1,cloud2,cloud3
I_sAddress	STRING(255)	Address of the structure / data point	e. g.: 'processdatamaster/temperature'

Outputs

61771

Output	Data type	Description	Possible values
O_xError	BOOL	error indication	- FALSE: no error - TRUE: error
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed

9.1.2 HMI_CloudConnect

61854

POC (STRUCT)

61855

Name	Data type	Description	Possible values
▪ X	DINT	Position of the contact point (X-coordinate value)	0...1280
▪ Y	DINT	Position of the contact point (Y-coordinate value)	0...480
▪ Pressed	BOOL	Contact status of the touch display	▪ FALSE: no contact ▪ TRUE: contact

FB_DataPointsTable (FB)

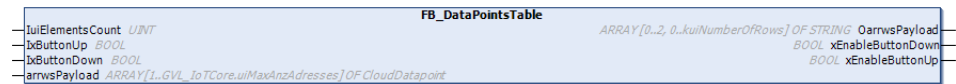
61857

Function block type: Function block

Behaviour model: --

Library: ifm_iiot.library

Symbol in CODESYS:



Description

61858

The function block is necessary for the presentation and logic of the visualisation side of the data points of the individual cloud systems.

Inputs

61859

Name	Data type	Description	Possible values
luiElementsCount	UINT	Number of elements	
lxButtonUp	BOOL	Input for "High" button	- FALSE: not pressed - TRUE: pressed
lxButtonDown	BOOL	Input for "Down" button	- FALSE: not pressed - TRUE: pressed
arrwsPayload	ARRAY of WSTRING	array with data points	per line: CloudDatapoint (STRUCT) (→ p. 44)

Outputs

61860

Name	Data type	Description	Possible values
OarrwsPayload	Array[] of STRING	Lines of the table to be displayed	
xEnableButtonDown	BOOL	Control output for "Down" button	- FALSE: do not activate - TRUE: activate
xEnableButtonUp	BOOL	Control output for "Up" button	- FALSE: do not activate - TRUE: activate

FB_DeviceTable (FB)

61862

Function block type: Function block (FB)

Behaviour model: --

Library: ifm_iod.library

Symbol in CODESYS:



Description

61863

The function block supports the visualisation and logic of the device table and the device configuration.

Inputs

61864

Input	Data type	Description	Possible values
IdiDevicesOnline	UINT	Number of the online devices in Visu	
IdiDevicesOffline	UINT	Number of the offline devices in Visu	
IO_x_MonitorSensors	BOOL	Triggering the monitoring function	- FALSE: Monitoring off - TRUE: Monitoring on

Outputs

61865

Output	Data type	Description	Possible values
OarrOfRows	ARRAY of IoTDevice	Structure with IoT-enabled devices for table	per line: IoTDevice (STRUCT) (→ p. 45)
OxLimitReached	BOOL	Error indication when memory limit is reached	- FALSE: not reached - TRUE: reached
OdiElementsCount	UINT	Number of elements in the visualisation	
OxDelRisingEdge	BOOL	Display when "Delete" button has been pressed	- FALSE: not pressed - TRUE: pressed
OxOnOffboardPressed	BOOL	Display when onboarding / offboarding has been pressed	- FALSE: not pressed - TRUE: pressed
OdiOnOffboardElement	UINT	Index of the item for which onboarding / offboarding has been pressed	
OxAddManualDevice	BOOL	Display if a device is to be added manually	- FALSE: do not add - TRUE: add
OmanDeviceToAdd	IoTDevice	Device to be added manually	IoTDevice (STRUCT) (→ p. 45)
OiLockScreen	INT	Position of the grey lock screen	
OxButtonEthSet	BOOL	Display when button for IP settings has been pressed	- FALSE: not pressed - TRUE: pressed

FB_flyingNumber (FB)

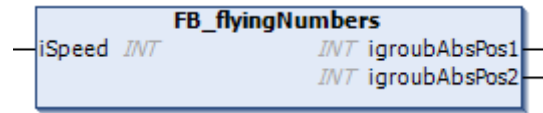
61867

Function block type: Function block

Behaviour model: --

Library: ifm_iot.library

Symbol in CODESYS:



Description

61868

The function block visualises the data transmission (flying numbers).

Inputs

61869

Name	Data type	Description	Possible values
iSpeed	INT	Speed of the numbers	0...10

Outputs

61870

Output	Data type	Description	Possible values
igroupAbsPos1	INT	Position of group 1	
igroupAbsPos2	INT	Position of group 2	

FB_Move (FB)

61872

Function block type: Function block

Behaviour model: --

Library: ifm_iot.library

Symbol in CODESYS:



Description

61873

The function block controls the movement of elements in the visualisation.

Inputs

61874

Input	Data type	Description	Possible values
r_Move_Value	REAL	Realistic movement distance	
r_Move_Velo	REAL	Movement acceleration	
i_Max_Move	INT	Maximum movement frequency, depending on number of images	0: Reverse movement >0: Frequency
r_Move_Var	REAL	Visualisation variable	
x_Start	BOOL	Input for movement start	- FALSE: Stop - TRUE: Start

Outputs

61875

Output	Data type	Description	Possible values
x_Move	BOOL	shows whether a movement is taking place	- FALSE: no movement - TRUE: movement

9.1.3 Cloud

61876

PRG_CloudConnect (PRG)

61877

Description

61878

The program block establishes the connection to the selected cloud systems.

CloudRemData (STRUCT)

61879

Name	Data type	Description	Possible values
▪ AnyVizData	RemAnyViz	Non-volatile data for AnyViz cloud	RemAnyViz (STRUCT) (→ p. 77)
▪ awsData	RemAWS	Non-volatile data for aws cloud	RemAWS (STRUCT) (→ p. 77)
▪ AzureData	RemAzure	Non-volatile data for Azure cloud	RemAzure (STRUCT) (→ p. 77)

RemAnyViz (STRUCT)

61880

Name	Data type	Description	Possible values
▪ uiProjectId	UINT	ID of the AnyViz project	e. g. 12345
▪ sPassword	STRING	Authenticated password for AnyViz	e. g. 'password'
▪ uiValueInterval	UINT	Update interval (value in s)	Initial value: 30

RemAWS (STRUCT)

61881

Name	Data type	Description	Possible values
▪ sClientId	STRING	Client ID	e. g. 'TestDevice1234'
▪ sHttpsHostname	STRING	Host name	e. g. 'xyz1234.iot.eu-central-1.amazonaws.com'
▪ uiValueInterval	UINT	Update interval (value in s)	Initial value: 60
▪ xConnect	BOOL	Permanent connection to the cloud	▪ FALSE: no ▪ TRUE: yes

RemAzure (STRUCT)

61882

Name	Data type	Description	Possible values
sFileName	STRING	Name of the SaS text file in the root directory of the USB stick	e. g. 'AzureSAS.txt'
xConnect	BOOL	Permanent connection to the cloud	▪ FALSE: no ▪ TRUE: yes
uiValueInterval	UINT	Update interval (value in s)	Initial value: 60
sSubDomainName	STRING(100)	Name of the Azure IoT hub	e. g. 'ifmRSTIoTHub'
sDeviceID	STRING(100)	Device ID	e. g. 'TestDevice'
wsDeviceSaS	WSTRING(255)	SaS Token of the device for authentication (necessary for c2d and d2c messages)	

FB_CloudGetJSONVal (FB)

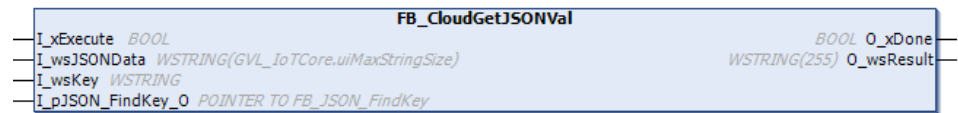
61884

Function block type: Function block

Behaviour model: EXECUTE

Library: ifm_iot.library

Symbol in CODESYS:



Description

61885

The function block reads a JSON key and outputs the value as a WSTRING.

Inputs

61886

Name	Data type	Description	Possible values
I_xExecute	BOOL	FB control	- FALSE: do not execute FB - TRUE: execute FB
I_wsJSONData	WSTRING	JSON string	
I_wsKey	WSTRING	Key	
I_pJSON_FindKey_O	POINTER	pointer to FB_JSON_FindKey	

Outputs

61887

Name	Data type	Description	Possible values
O_xDone	BOOL	indication whether FB has been successfully processed	- FALSE: FB will be processed - TRUE: FB processing completed
O_wsResult	WSTRING	Returned value	

9.2 Library ifmDevice_ecomatDisplay.library

Content

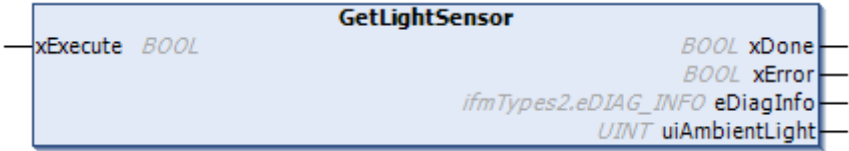
GetLightSensor.....	80
GetLightSensorCyclic	82
GetTemperature	84
GetTemperaturesCyclic.....	86
LCDcontrol.....	88
GetBootupBacklight.....	90
GetLCD_Backlight	92
SetBootupBacklight	94
SetLCD_Backlight	96

61981

9.2.1 GetLightSensor

57920

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmDevice_ecomatDisplay.library
Symbol in CODESYS:



Description

57921

The function block returns the light intensity value of the integrated light sensor.

-
- Only one instance of the FB may be active within an application. All function block calls will be cancelled, and an error message is given.

 - Only call up one instance of the function block within the application!

Input parameters

57785

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE ⇒ TRUE	FB is executed once
			Other	No impact on FB processing

Output parameters

57922

Parameter	Data type	Description	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	▪ FB successfully executed ▪ FB can be called again
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	▪ Error occurred ▪ Action could not be executed ▪ Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
uiAmbientLight	UINT	Value of the light intensity determined by the integrated light sensor	0...4096 = 0...100%	

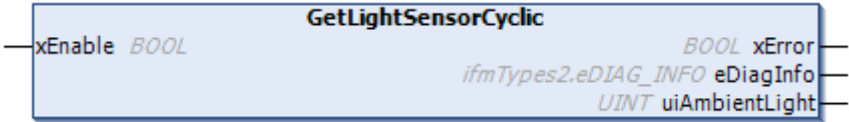
Diagnostic codes:

- STAT_INACTIVE State: FB/Function is inactive.
- STAT_BUSY State: FB/Function is currently executed.
- STAT_DONE State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- ERR_INSTANCE_LIMIT Fehler: Mehr als eine Instanz des FB erzeugt; diese FB-Instanz wird nicht ausgeführt
- ERR_READ_LIGHT_SENSOR
 - error while reading the light sensor value.
 - ▶ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
 - ▶ Contact the ifm Service Center!

9.2.2 etLightSensorCyclic

57924

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmDevice_ecomatDisplay.library
Symbol in CODESYS:



Description

57925

The function block returns the light intensity value of the integrated light sensor in a cyclical interval of 150 ms.



Only one instance of the FB may be active within an application. All function block calls will be cancelled, and an error message is given.

- Only call up one instance of the function block within the application!

Input parameters

57926

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Disable function block
			TRUE	Enable function blocks

Output parameters

57927

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	- Error occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
uiAmbientLight	UINT	Value of the light intensity determined by the integrated light sensor	0...4096 = 0...100%	

Diagnostic codes:

- STAT_INACTIVE State: FB/Function is inactive.
- STAT_PREPARING State: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
- STAT_DONE State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- ERR_INSTANCE_LIMIT Fehler: Mehr als eine Instanz des FB erzeugt; diese FB-Instanz wird nicht ausgeführt
- ERR_READ_LIGHT_SENSOR
 - error while reading the light sensor value.
 - ▶ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
 - ▶ Contact the ifm Service Center!

9.2.3 GetTemperature

57937

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmDevice_ecomatDisplay.library
Symbol in CODESYS:



Description

57938

The function block reads the following system temperatures:

- PCB temperature
- Processor temperature



Only one instance of the FB may be active within an application. All function block calls will be cancelled, and an error message is given.
► Only call up one instance of the function block within the application!

Input parameters

57785

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE ⇒ TRUE	FB is executed once
			Other	No impact on FB processing

Output parameters

57939

Parameter	Data type	Description	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	- FB successfully executed - FB can be called again
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	- Error occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
stTemperatures	stSYSTEM_TEMPERATURES	Structure with the system temperatures	→ stSYSTEM_TEMPERATURES (STRUCT)	

Diagnostic codes:

- STAT_INACTIVE State: FB/Function is inactive.
- STAT_BUSY State: FB/Function is currently executed.
- STAT_DONE State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- ERR_INSTANCE_LIMIT Fehler: Mehr als eine Instanz des FB erzeugt; diese FB-Instanz wird nicht ausgeführt
- ERR_READ_TEMPERATURE_CORE_0 Error while reading the processor temperature
 - ▶ Contact the ifm Service Center!
- ERR_READ_TEMPERATURE_BOARD Error while reading the PCB temperature.
 - ▶ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
 - ▶ Contact the ifm Service Center!

9.2.4 GetTemperaturesCyclic

57941

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmDevice_ecomatDisplay.library

Symbol in CODESYS:



Description

57942

The function block reads the following system temperatures cyclically at an interval of 2000 ms:

- PCB temperature
- Processor temperature



Only one instance of the FB may be active within an application. All function block calls will be cancelled, and an error message is given.

- Only call up one instance of the function block within the application!

Input parameters

57926

Parameter	Data type	Description	Possible values	
xEnable	BOOL	Control activity of the FB	FALSE	Disable function block
			TRUE	Enable function blocks

Output parameters

57943

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	- Error occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
stTemperatures	stSYSTEM_TEMPERATURES	Structure with the system temperatures	→ stSYSTEM_TEMPERATURES	

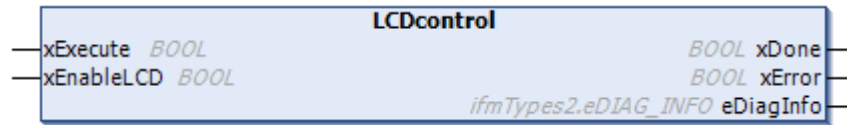
Diagnostic codes:

- STAT_INACTIVE State: FB/Function is inactive.
- STAT_PREPARING State: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
- STAT_DONE State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- ERR_INSTANCE_LIMIT Fehler: Mehr als eine Instanz des FB erzeugt; diese FB-Instanz wird nicht ausgeführt
- ERR_READ_TEMPERATURE_CORE_0 Error while reading the processor temperature
 - ▶ Contact the ifm Service Center!
- ERR_READ_TEMPERATURE_BOARD Error while reading the PCB temperature.
 - ▶ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
 - ▶ Contact the ifm Service Center!

9.2.5 LCDcontrol

57894

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmDevice_ecomatDisplay.library
Symbol in CODESYS:



Description

57895

The function block switches on/off the LCD display and the background illumination.



Only one instance of the FB may be active within an application. All function block calls will be cancelled, and an error message is given.

- Only call up one instance of the function block within the application!

Input parameters

57896

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE ⇔ TRUE	FB is executed once
			Other	No impact on FB processing
xEnableLCD	BOOL	Switching on /off LCD (low power mode).	FALSE	Switch on LCD.
			TRUE	Switch off LCD.

Output parameters

57898

Parameter	Data type	Description	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	▪ FB successfully executed ▪ FB can be called again
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	▪ Error occurred ▪ Action could not be executed ▪ Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

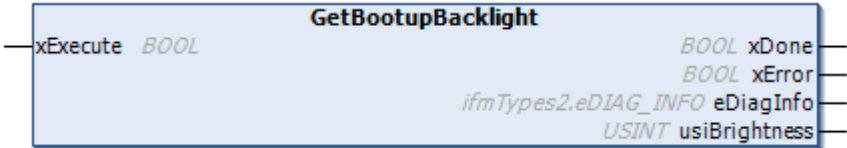
Diagnostic codes:

- STAT_INACTIVE State: FB/Function is inactive.
- STAT_BUSY State: FB/Function is currently executed.
- STAT_DONE State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- ERR_INSTANCE_LIMIT Fehler: Mehr als eine Instanz des FB erzeugt; diese FB-Instanz wird nicht ausgeführt
- ERR_INTERNAL Error: Internal system error
 - ▶ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
 - ▶ Contact the ifm Service Center!

9.2.6 GetBootupBacklight

57900

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmDevice_ecomatDisplay.library
Symbol in CODESYS:



Description

57901

The function block reads the set LCD brightness settings for the boot process.



Only one instance of the FB may be active within an application. All function block calls will be cancelled, and an error message is given.
► Only call up one instance of the function block within the application!

Input parameters

57785

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE ⇒ TRUE	FB is executed once
			Other	No impact on FB processing

Output parameters

57902

Parameter	Data type	Description	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	▪ FB successfully executed ▪ FB can be called again
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	▪ Error occurred ▪ Action could not be executed ▪ Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
usiBrightness	USINT	Set brightness value for the device boot process	0...100 %	

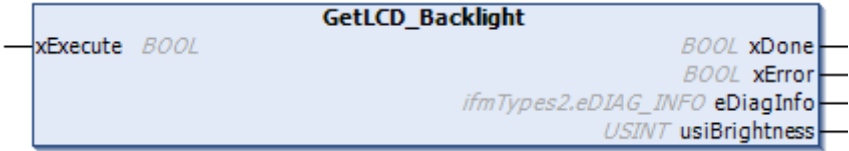
Diagnostic codes:

- STAT_INACTIVE State: FB/Function is inactive.
- STAT_BUSY State: FB/Function is currently executed.
- STAT_DONE State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- ERR_INSTANCE_LIMIT Fehler: Mehr als eine Instanz des FB erzeugt; diese FB-Instanz wird nicht ausgeführt
- .
- ERR_GET_BOOTUP_LCD_BRIGHTNESS error while reading the brightness value
 - ▶ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
 - ▶ Contact the ifm Service Center!

9.2.7 GetLCD_Backlight

57904

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmDevice_ecomatDisplay.library
Symbol in CODESYS:



Description

57905

The function block reads the set LCD brightness settings for normal display operation.

- 
- Only one instance of the FB may be active within an application. All function block calls will be cancelled, and an error message is given.

 - Only call up one instance of the function block within the application!

Input parameters

57785

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE ⇒ TRUE	FB is executed once
			Other	No impact on FB processing

Output parameters

57906

Parameter	Data type	Description	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	▪ FB successfully executed ▪ FB can be called again
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	▪ Error occurred ▪ Action could not be executed ▪ Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
usiBrightness	USINT	Set brightness value for normal device operation	0...100 %	

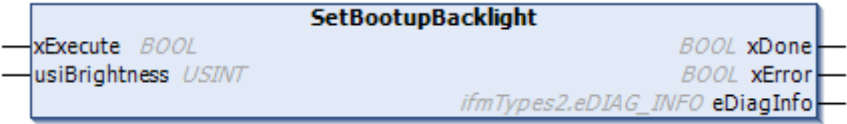
Diagnostic codes:

- STAT_INACTIVE State: FB/Function is inactive.
- STAT_BUSY State: FB/Function is currently executed.
- STAT_DONE State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- ERR_INSTANCE_LIMIT Fehler: Mehr als eine Instanz des FB erzeugt; diese FB-Instanz wird nicht ausgeführt
- ERR_GET_LCD_BRIGHTNESS error while reading the brightness value
 - ▶ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
 - ▶ Contact the ifm Service Center!

9.2.8 SetBootupBacklight

57908

Function block type: Function block (FB)
Behaviour model: EXECUTE
Library: ifmDevice_ecomatDisplay.library
Symbol in CODESYS:



Description

57909

The function block sets the value for the LCD brightness for the boot process.



Only one instance of the FB may be active within an application. All function block calls will be cancelled, and an error message is given.
► Only call up one instance of the function block within the application!

Input parameters

57910

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE ⇒ TRUE	FB is executed once
			Other	No impact on FB processing
usiBrightness	USINT	Brightness value for the device boot process	0...100 % Default value = 20: 80 %	

Output parameters

57911

Parameter	Data type	Description	Possible values	
xDone	BOOL	Indication of whether execution of the FB has been successfully completed	FALSE	FB is executed
			TRUE	▪ FB successfully executed ▪ FB can be called again
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	▪ Error occurred ▪ Action could not be executed ▪ Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	

Diagnostic codes:

- STAT_INACTIVE State: FB/Function is inactive.
- STAT_BUSY State: FB/Function is currently executed.
- STAT_DONE State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- ERR_INSTANCE_LIMIT Fehler: Mehr als eine Instanz des FB erzeugt; diese FB-Instanz wird nicht ausgeführt
- ERR_INVALID_VALUE Error: value on the input parameter usiBrightness outside of the value range.
- ERR_SET_BOOTUP_LCD_BRIGHTNESS error while setting the brightness value.
 - ▶ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
 - ▶ Contact the ifm Service Center!

9.2.9 SetLCD_Backlight

57914

Function block type: Function block (FB)

Behaviour model: ENABLE

Library: ifmDevice_ecomatDisplay.library

Symbol in CODESYS:



Description

57915

The function block sets the value for the LCD brightness for normal operation.



Only one instance of the FB may be active within an application. All function block calls will be cancelled, and an error message is given.

- Only call up one instance of the function block within the application!

Input parameters

57916

Parameter	Data type	Description	Possible values	
xExecute	BOOL	Control execution of the FB	FALSE ⇒ TRUE	FB is executed once
			Other	No impact on FB processing
usiBrightness	USINT	Brightness value for normal device operation	0...100 [%] Default value = 20: 80 [%] Background illumination off = 0 [%]	

Output parameters

57917

Parameter	Data type	Description	Possible values	
xError	BOOL	Indication if an error occurred during the FB execution	FALSE	No error occurred or the FB is still being executed
			TRUE	- Error occurred - Action could not be executed - Note diagnostic information
eDiagInfo	DIAG_INFO	Diagnostic information	→ List below (diagnostic codes)	
xPrepared	BOOL	State of the FB outputs	FALSE	FB is still processed; FB outputs invalid
			TRUE	FB has been processed; FB outputs valid

Diagnostic codes:

- STAT_INACTIVE State: FB/Function is inactive.
- STAT_PREPARING State: FB/FUN is processed; final results are not yet available. Some output values are updated in each PLC cycle.
- STAT_DONE State: FB/Function has been successfully executed and completed. There are valid results on the outputs.
- ERR_INSTANCE_LIMIT Fehler: Mehr als eine Instanz des FB erzeugt; diese FB-Instanz wird nicht ausgeführt
- ERR_INVALID_VALUE Error: value on the input parameter usiBrightness outside of the value range.
- ERR_SET_LCD_BRIGHTNESS error while setting the brightness value.
 - ▶ Contact the ifm Service Center!
- ERR_UNDEFINED Error: Unknown error
 - ▶ Contact the ifm Service Center!