EUCHNER

Application

EKS2 – Integration of EKS2 with PROFINET Interface in BECKHOFF TwinCAT 3

EN

Contents

1. About this document

1.1. Version

Version	Date	Change/addition	Chapter
01-05/26	05/2026	Prepared	All

1.2. Scope

This application describes the integration and configuration of the Electronic-Key-System EKS2 (hereinafter abbreviated to *EKS2*) with PROFINET-IO interface in Beckhoff TwinCAT 3.

1.3. Target group

Design engineers and installation planners for safety devices on machines as well as setup and servicing staff possessing the following expertise:

- specialist knowledge in handling safety components
- expertise in the installation, setup, programming and diagnostics of programmable logic controllers (PLC) and bus systems
- knowledge about the applicable EMC regulations
- knowledge about the applicable regulations on operational safety and accident prevention.

1.4. Supplementary documents

The overall documentation for this application consists of the following documents:

Document title (document number)	Contents
Operating instructions (MAN20001715)	Transponder-coded Electronic-Key-System Electronic-Key-System EKS2
Possibly enclosed data sheets	Item-specific information about deviations or additions

1.5. Notice

This document is based on the operating instructions for the EKS2. You can refer to the operating instructions for technical details and more information.

EN

2. Components/modules used

2.1. EUCHNER

Description	Order number / item
Evaluation unit/read unit EKS2	170904 / EKS2-E-PN-M-H1-170904
	170915 / EKS2-R-I1A1-B1-170915
Electronic-Key EKS2	168433 / EKS2-K-K-B-D2-BK-168433
	168434 / EKS2-K-K-B-D2-BU-168434
	168435 / EKS2-K-K-B-D2-GN-168435
	168438 / EKS2-K-K-B-D2-OG-168438
	168432 / EKS2-K-K-B-D2-RD-168432
	168437 / EKS2-K-K-B-D2-WH-168437
	168436 / EKS2-K-K-B-D2-YE-168436

TIP!

More information and downloads for the aforementioned EUCHNER products can be found at www.euchner.com. Simply enter the order number in the search box.

2.2. Others

Description	Order number / item
CX2030-M930	CX2030-M930

2.3. Software

Description	Version
TwinCAT	v3.1.4024.62

3. Functional description

3.1. General

This application supports integration of the EKS2. The description covers hardware integration, as well as integration and use of the optional software library for data preparation.

The EKS2 operates as an IO device on the PROFINET network.

The EKS2 operating instructions contain the communication data.

4. PROFINET configuration

The EKS2 exchanges the data in the PROFINET environment via data blocks with the associated communication data. The configuration software uses the GSDML file for configuration and parameterization of the required modules.

4.1. Integrating the GSDML file

A GSD file in GSDML format is required for integrating the EKS2 in TwinCAT 3. The GSD files are available in the Downloads area at www.euchner.com. Always use the current GSD file. Copy the GSDML file and the associated bitmap file to the corresponding TwinCAT directory.

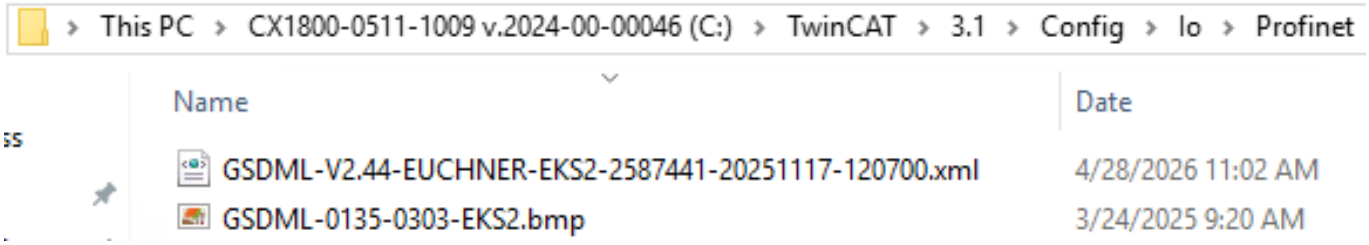


Fig. 1: Adding GSD file

4.2. Setting the control system parameters

The PROFINET cycle time of the PLC task depends on the application's requirements. More information is available in the BECKHOFF Information System.

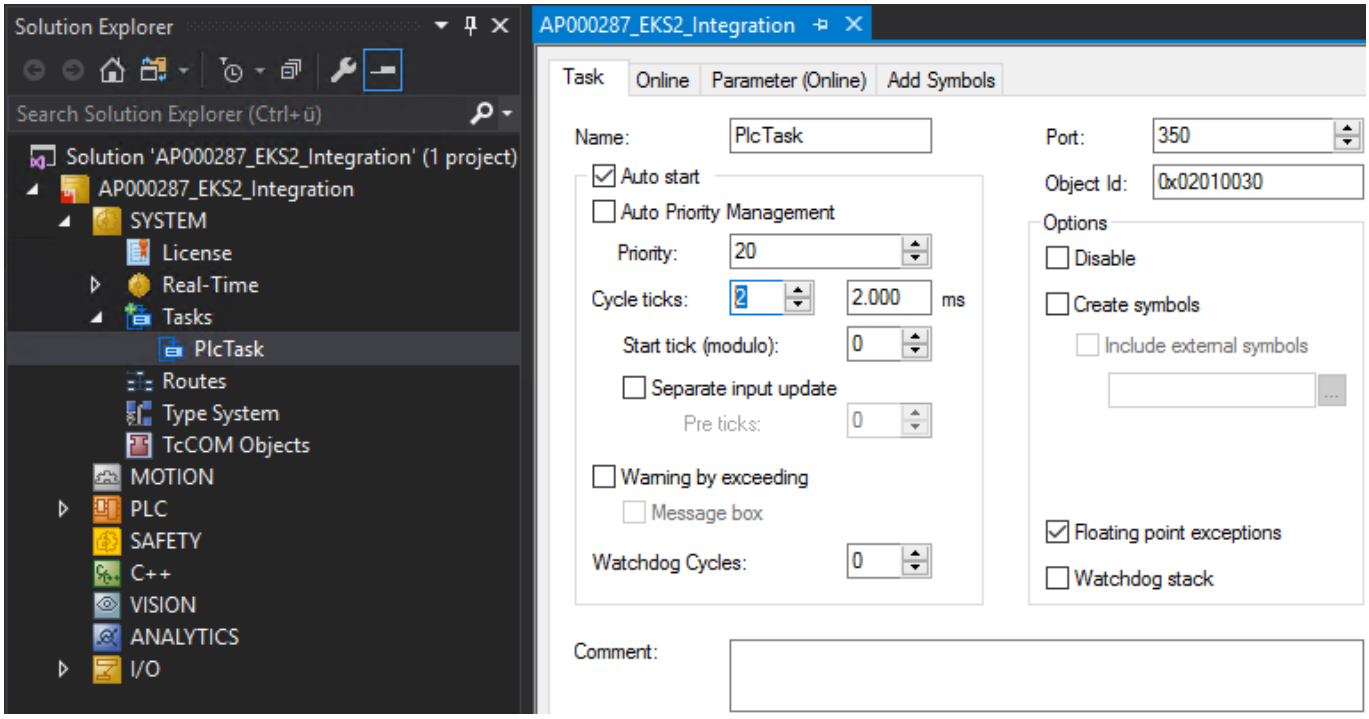


Fig. 2: Setting the control system parameters

EN

4.3. Adding the PROFINET controller

Setting up the PROFINET controller

1. Right-click Devices under I/O in the Solution Explorer and select the Scan function.

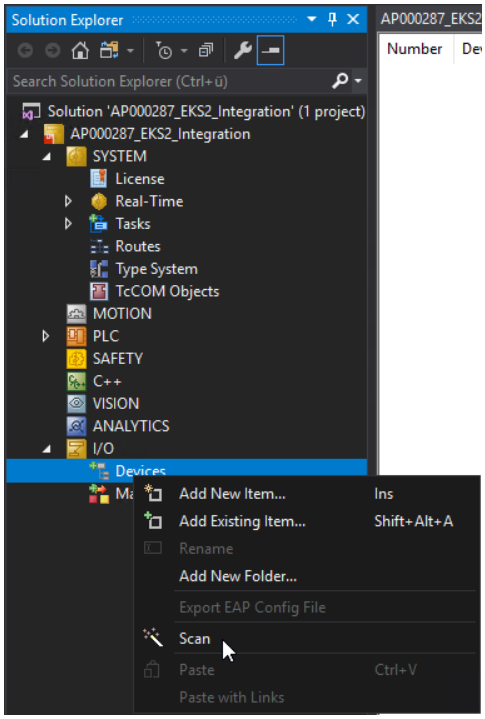


Fig. 3: Adding PROFINET network

NOTICE!

TwinCAT must be in Config mode for scan processes.

2. Select the PROFINET controller and confirm with OK.

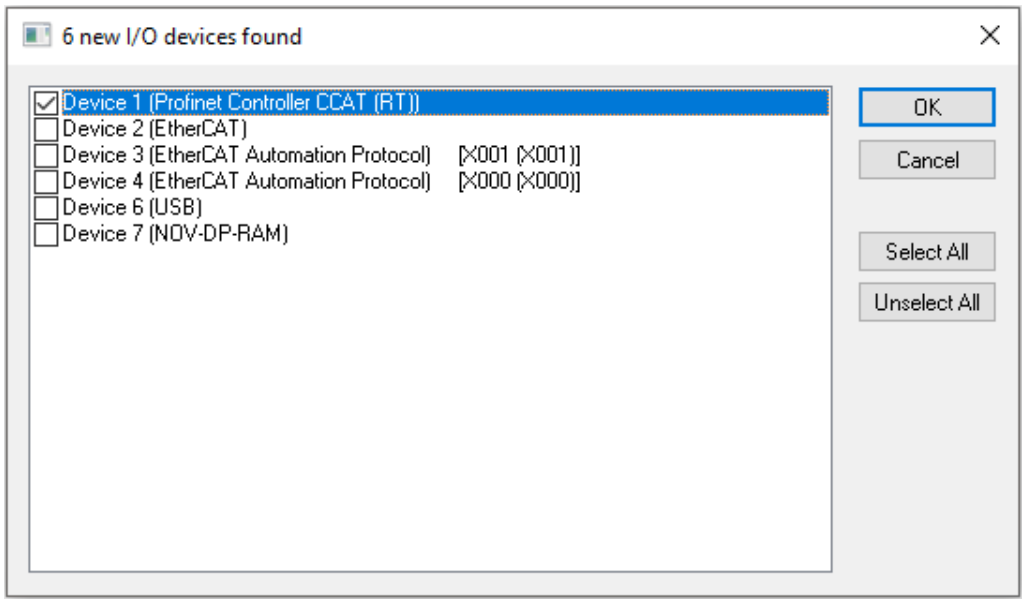


Fig. 4: Selection of PROFINET controller

3. Answer *No* to the prompt to search for additional devices automatically. Unambiguous assignment of the EKS2 is not ensured during the automatic search.

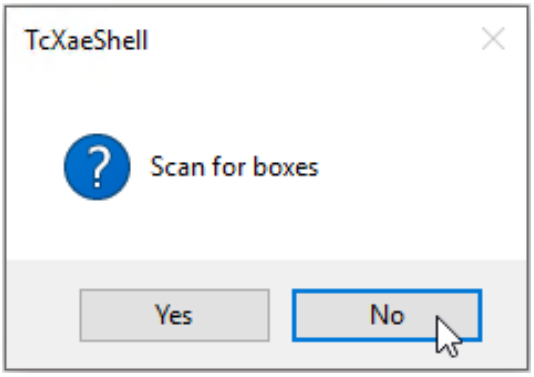


Fig. 5: Rejecting search for devices

4.4. Adding the EKS2 with PROFINET interface

1. Right-click the PROFINET controller and select the *Add New Item...* function.

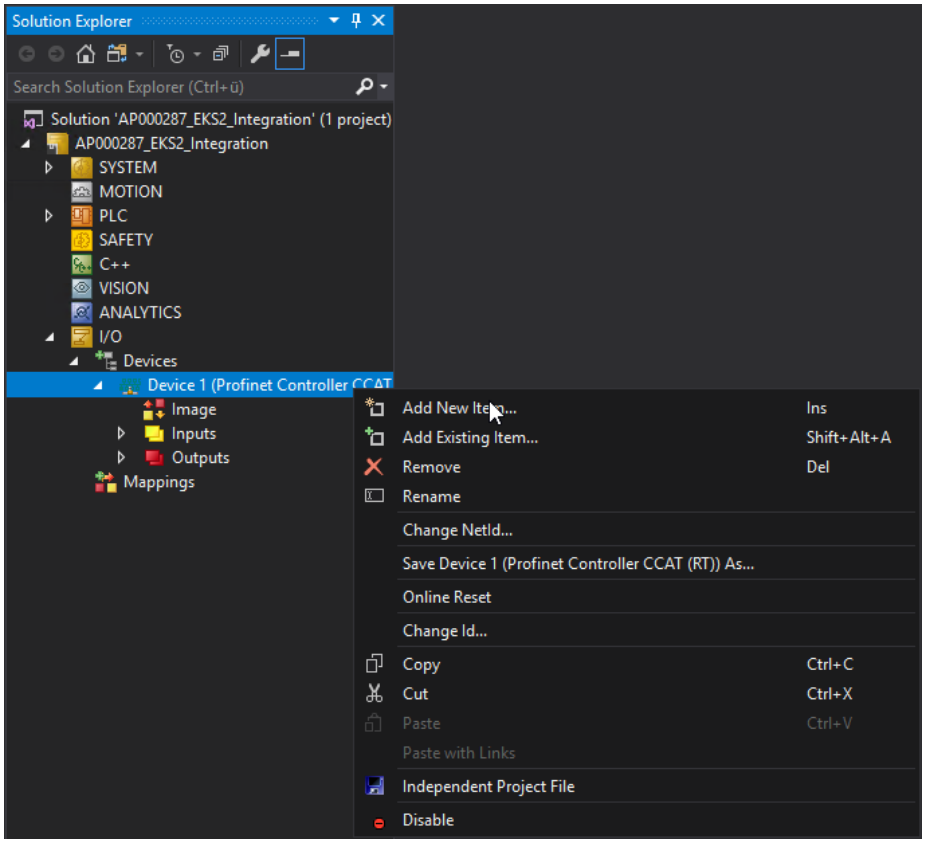


Fig. 6: Adding a device

EN

2. Select the corresponding GSDML file.

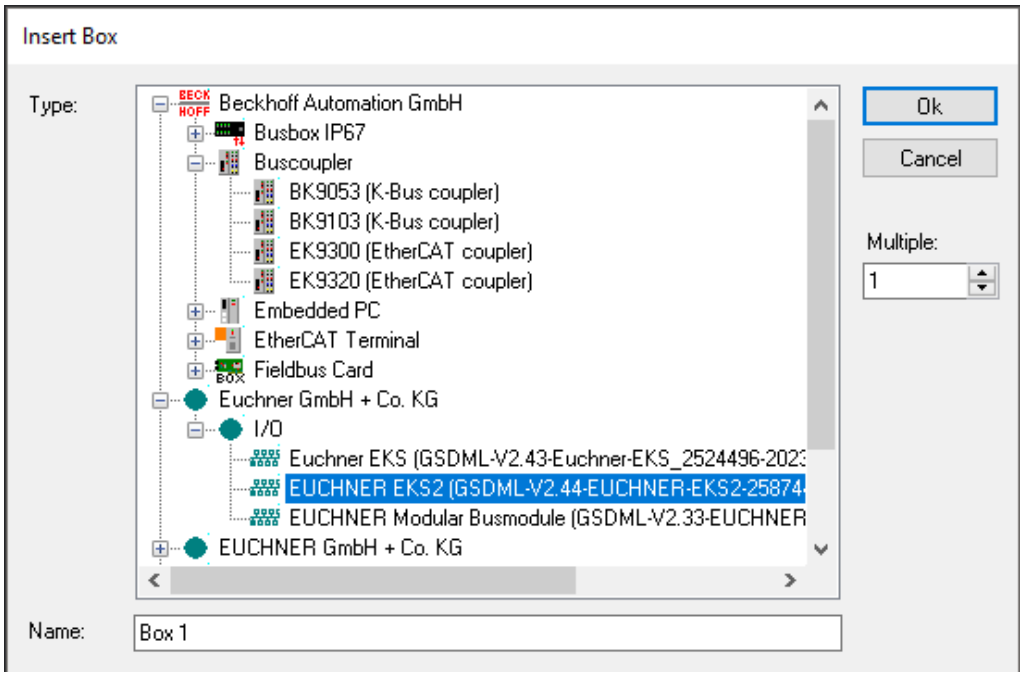


Fig. 7: Selection of the GSDML file

4.5. Parameterization of the EKS2

4.5.1. Device name and IP address

The following configuration specifies the PROFINET parameters. For this purpose, double-click to open the properties of the EKS2 device in the *Solution Explorer*. Then select the Device tab.

- Device name
- IP address: fixed

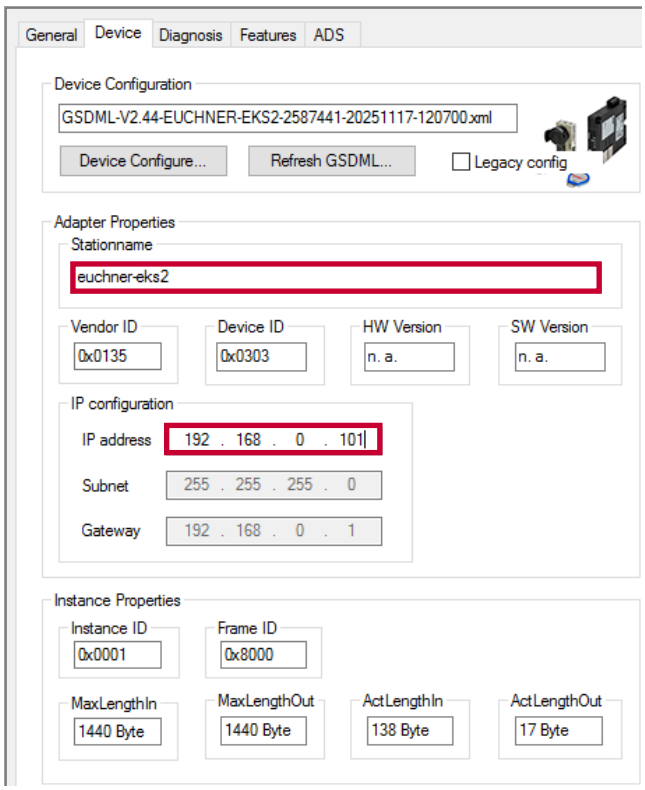


Fig. 8: PROFINET parameters

4.6. Modules and submodules

The following modules are available and are either already included in the GSDML for the evaluation unit as a standard configuration or, optionally, can be added:

Module	Parameter [default value]	Description	Slot	
EKS2 Diagnose Basic	Alarm setting: Activated/Deactivated [Deactivated] Data format setting: Little/Big Endian [Big Endian]	‣ Reading status messages ‣ Parameterizing the alarms ‣ Parameterizing the data format	1	
EKS2 Diagnose Extended	Alarm setting: Activated/Deactivated [Deactivated] Data format setting: Little/Big Endian [Big Endian]	‣ Reading status and diagnostic messages ‣ Parameterizing the alarms ‣ Parameterizing the data format	1	
Read: Transponder UID Standard mode with 8 bytes		‣ Activating standard mode, see operating instructions ‣ Reading the transponder's unique serial number (UID)	2	
Read: Transponder UID-only mode with 8 bytes		‣ Activating UID-only mode, see operating instructions ‣ Reading the transponder's unique serial number (UID) ‣ Selection of operating mode not possible	2	
Read: EU001 Base Module	Automatic data evaluation: Yes/No [Yes]	Activating automatic data evaluation	3	3.1
	Company: [0] Plant: [0] Department: [0]	These fields are evaluated one after the other as a hierarchy.		
	Cost center: 1 - 5 [0]	These fields are combined with an OR operator		
	Exception value: 1, 2 [0]	The values in these fields are superordinate to the other fields in the evaluation.		
	Use expiry date: Yes/No [No]	‣ Internal evaluation of the transponder's expiry date ‣ The current date additionally must be defined in the control system and sent to the evaluation unit via an acyclical command.		
	Automatic Electronic-Key management: Activated/Deactivated [Deactivated]	Activating automatic Electronic-Key management		
Read: EU001 header data with 18 bytes		Additional reading of the header data in the control system		3.2
Read: EU001 User data with 16 bytes	Start address: [26] Number of bytes to be read: [dependent on the module]	Reading user data with 16 bytes [1]		3.3
Read: EU001 User data with 32 bytes		Reading user data with 32 bytes [1]		3.3
Read: EU001 User data with 64 bytes		Reading user data with 64 bytes [1]		3.3
Read: EU001 User data with 90 bytes		Reading user data with 90 bytes [1]		3.3
Read: EU000 Base Module	Use expiry date: Yes/No [No]	‣ Internal evaluation of the transponder's expiry date ‣ The current date additionally must be defined in the control system and sent to the evaluation unit via an acyclical command.	3	3.1
	Start address of the expiry date: [0]	Start address of the expiry date		
Read: EU000 User data with 16 bytes	Start address: [0] Number of bytes to be read: [dependent on the module]	Reading user data with 16 bytes		3.2
Read: EU000 User data with 32 bytes		Reading user data with 32 bytes		
Read: EU000 User data with 64bytes		Reading user data with 64 bytes		
Read: EU000 User data with 116 bytes		Reading user data with 116 bytes (default)		
Read/Write: EU001 MO Module	Machine group: 1-4 [1] Operating mode after system start: MO0-MO5 [MO0]	‣ Compatible only with EU001 Base Module ‣ Setting the machine group ‣ Configuring the parameters for selection of operating mode in the evaluation unit	4	
Read/Write: EU000 MO Module	Start address of the MO value: [0] Operating mode after system start: MO0-MO5 [MO0]	‣ Compatible only with EU000 Base Module ‣ Configuring the parameters for selection of operating mode in the evaluation unit		

1) The first four bytes contain the Personnel number.
The standard modules are in bold.

Table 1: EKS2 modules

EN

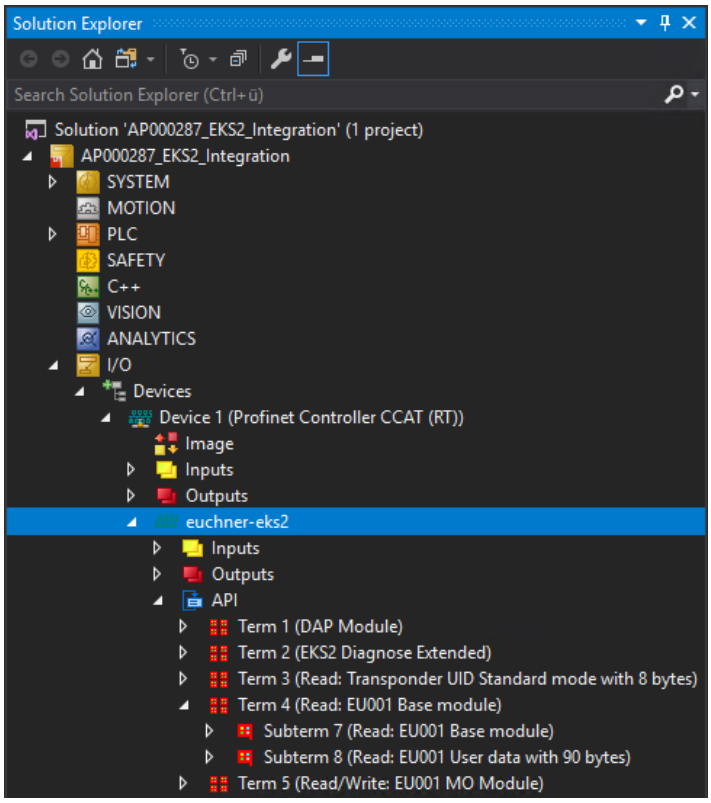


Fig. 9: Standard configuration

4.7. EU001 data structure

4.7.1. Configuration of a module

The optional *Read: EU001 header data with 18 bytes* submodule supplements the standard configuration to illustrate the configuration. The control system uses it to read the header data and display them. Applications without header data do not require the following explanations.

1. Open the properties of the EKS2 by double-clicking in the *Solution Explorer*. Then select the *Device* tab and call the *Device Configure...* function.

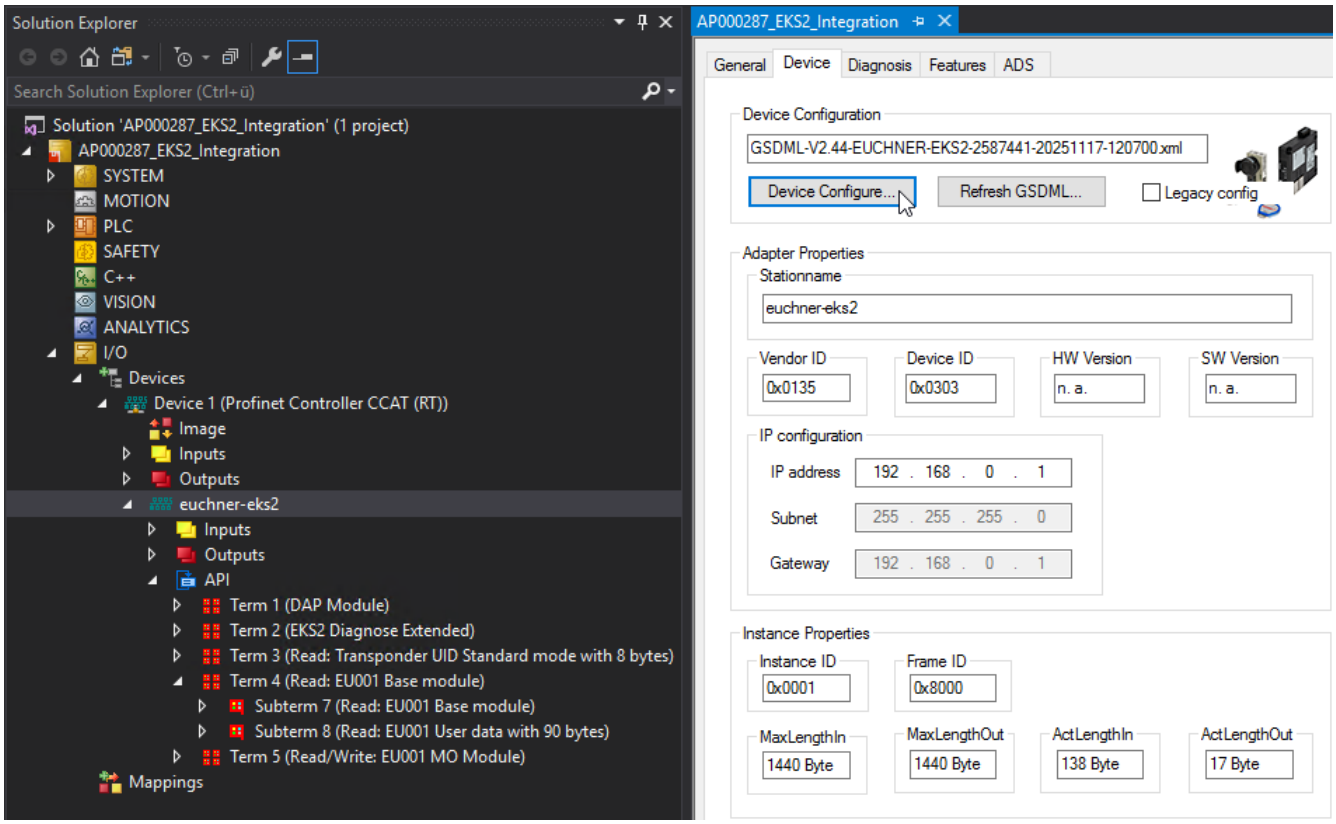


Fig. 10: Opening device configuration

EN

2. Select *slot 3* (1) on the left side. Then drag the *Read: EU001 header data with 18 bytes* module from the list on the lower right (2) into the free *subslot 2* in the middle (3).

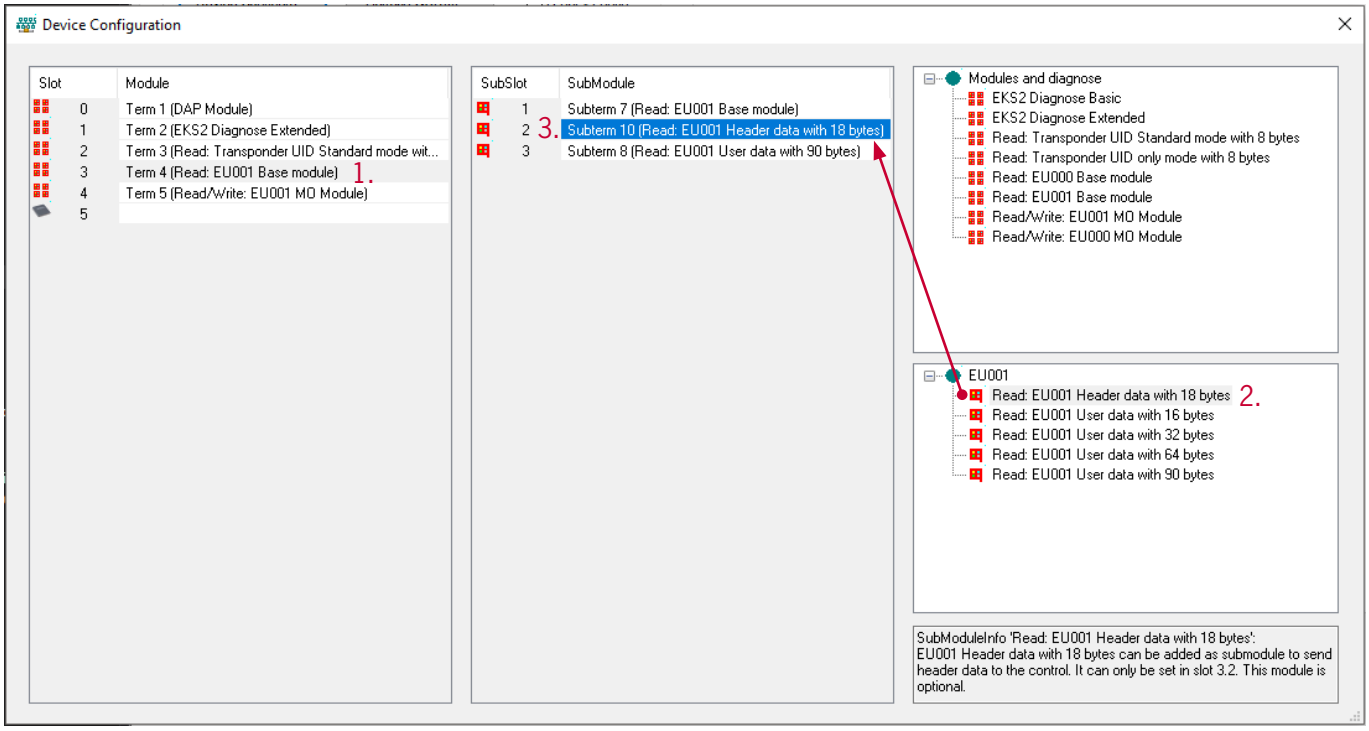


Fig. 11: Adding a submodule

4.7.2. Parameterization of the endianness

The endianness must be compatible with the control system used for correct interpretation of the data stored in the EKS2 Electronic-Key. This application uses a BECKHOFF control system with little endian format. Therefore, select the *Little Endian* setting.

Select the configured EKS2 device in the Solution Explorer.

Then expand the structure via *API –> Term x (EKS2 Diagnose Extended) –> Subterm x (EKS2 Diagnose Extended)*. Double-click to open the subterm. Then select the *Parameterize Module* tab.

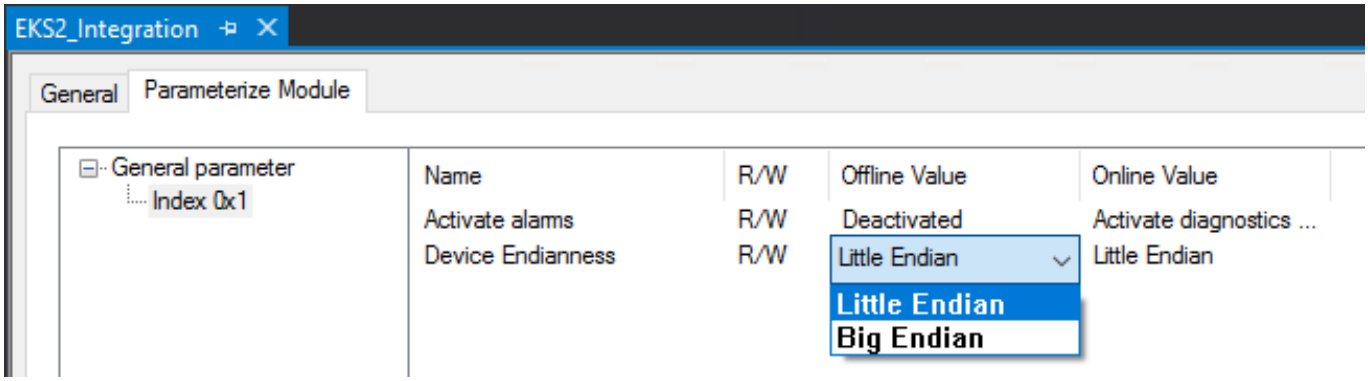


Fig. 12: Parameterization of the endianness

	NOTICE Parameters are transmitted exclusively via the *Activate Configuration* button.

EN

4.8. EU000 data structure

4.8.1. Configuration of the modules and submodules

In order to use the EU000 data structure, two preconfigured modules must be replaced with the associated EU000 modules.

1. Open the properties of the EKS2 by double-clicking in the *Solution Explorer*. Then select the *Device* tab and call the *Device Configure...* function.

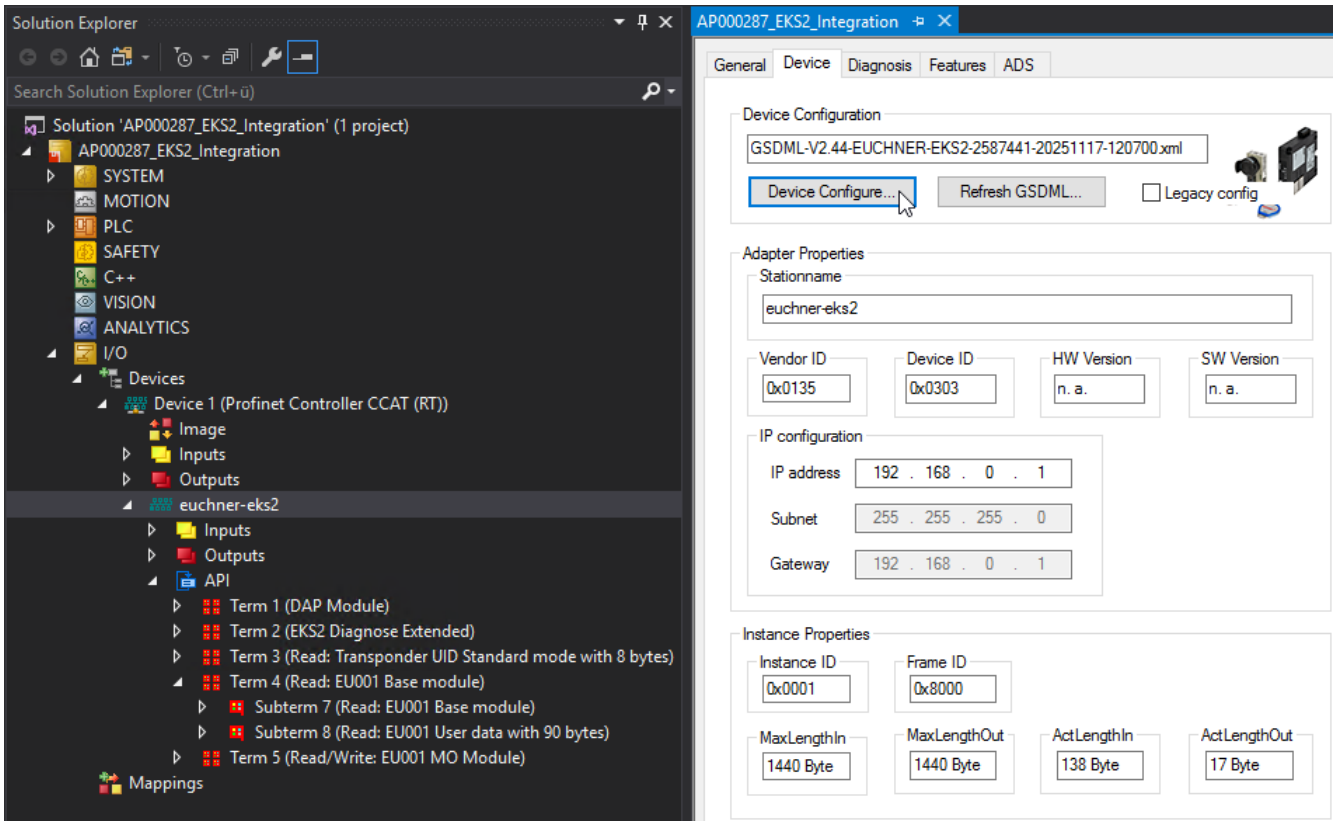


Fig. 13: Opening device configuration

2. Delete all preconfigured modules with the *EU001* designation on the left side. Then select the corresponding EU000 modules in the upper right window and assign them to the free slots on the left side.

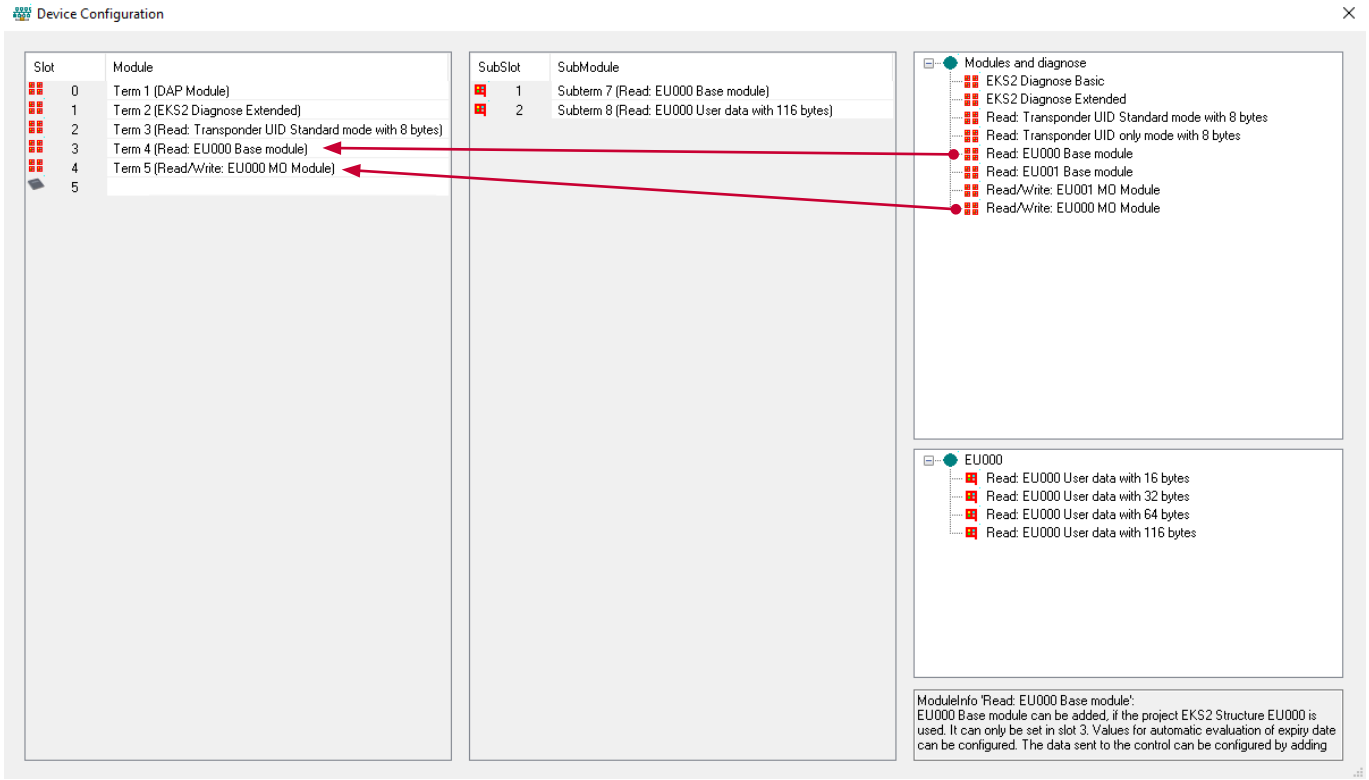


Fig. 14: Configured EU000 modules

	NOTICE The *Read: EU000 User data with 116 bytes* submodule in the *Read: EU000 Base module* module is already preconfigured. If fewer freely programmable bytes are required, the submodule can be replaced with smaller submodules.

	NOTICE The parameterization of the configured submodules depends on the application's requirements. *Table 1* contains the associated parameters.

EN

5. Using the BECKHOFF library

The library provides templates for simple integration of the EKS2 into the application.

The Applications area under the Downloads area at www.euchner.com contains the EKS2 library.

5.1. Installation of the library

1. Open the *Library Repository* on the *PLC* tab.
2. Then load the library from the corresponding directory using the *Install...* function.

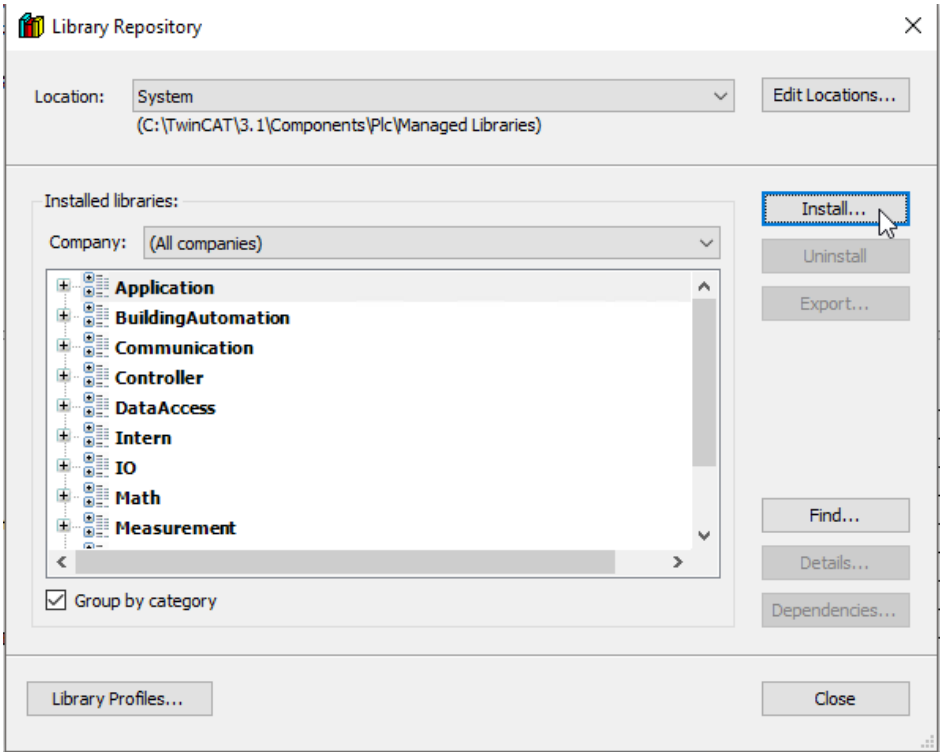


Fig. 15: Installing library

	IMPORTANT! Evaluation units EKS2 from **version 1.2.0** support the library.

	NOTICE As soon as the library is installed, it is displayed in (Miscellaneous).

3. In the *Solution Explorer*, right-click to add the library to the project via *References –> Add library...*.

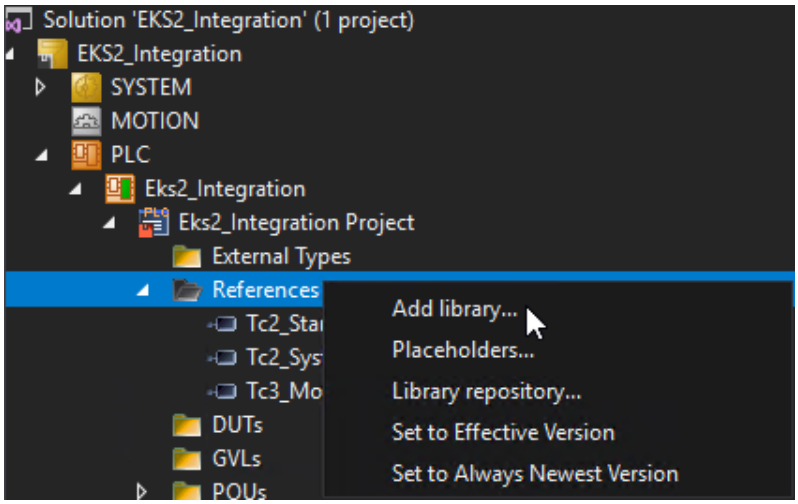


Fig. 16: Adding library to the project

4. Select the library provided by EUCHNER.

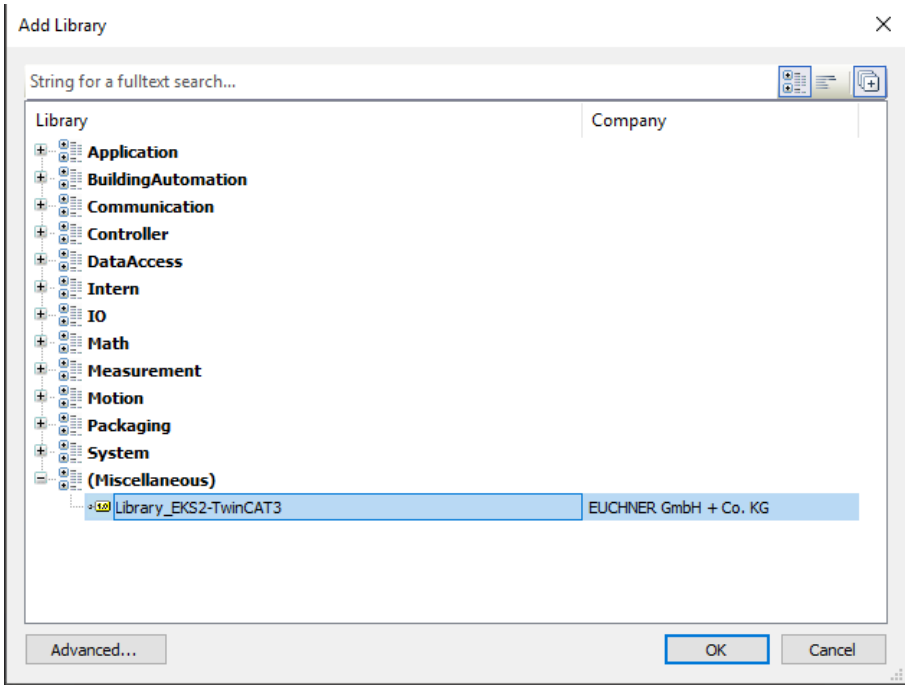


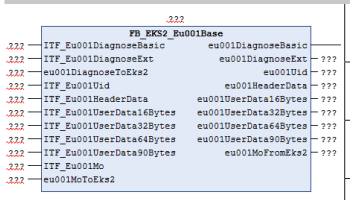
Fig. 17: Library of the library

EN

5.2. EU001 data structure

5.2.1. Description of the block interface

The following table describes the parameters of the *FB_EKS2_Eu001Base* block.



Variable	Use	Data type	Description
ITF_Eu001DiagnoseBasic	INPUT	ITF_EKS2_Eu001DiagnoseBasic	Interface for Diagnose Basic module
ITF_Eu001DiagnoseExt	INPUT	ITF_EKS2_Eu001DiagnoseExt	Interface for Diagnose Extended module
eu001DiagnoseToEks2	INPUT	ST_EKS2_Structure_Output_Eu001Diagnose	Output data for Diagnose (Base and Extended) module
ITF_Eu001Uid	INPUT	ITF_EKS2_Eu001Uid	Interface for UID module
ITF_Eu001HeaderData	INPUT	ITF_EKS2_Eu001HeaderData	Interface for Header Data module
ITF_Eu001UserData16Bytes	INPUT	ITF_EKS2_Eu001UserData16Bytes	Interface for User Data 16 Bytes module
ITF_Eu001UserData32Bytes	INPUT	ITF_EKS2_Eu001UserData32Bytes	Interface for User Data 32 Bytes module
ITF_Eu001UserData64Bytes	INPUT	ITF_EKS2_Eu001UserData64Bytes	Interface for User Data 64 Bytes module
ITF_Eu001UserData90Bytes	INPUT	ITF_EKS2_Eu001UserData90Bytes	Interface for User Data 90 Bytes module
ITF_Eu001Mo	INPUT	ITF_EKS2_Eu001Mo	Interface for MO module
eu001MoToEks2	INPUT	ST_EKS2_Structure_Output_Eu001Mo	Output data for MO module
eu001DiagnoseBasic	OUTPUT	ST_EKS2_Structure_Input_Eu001DiagnoseBasic	Diagnose Basic structured data
eu001DiagnoseExt	OUTPUT	ST_EKS2_Structure_Input_Eu001DiagnoseExt	Diagnose Extended structured data
eu001Uid	OUTPUT	ST_EKS2_Structure_Input_Eu001Uid	UID structured data
eu001HeaderData	OUTPUT	ST_EKS2_Structure_Input_Eu001HeaderData	Header Data structured data
eu001UserData16Bytes	OUTPUT	ST_EKS2_Structure_Input_Eu001UserData16Bytes	User Data 16 Bytes structured data
eu001UserData32Bytes	OUTPUT	ST_EKS2_Structure_Input_Eu001UserData32Bytes	User Data 32 Bytes structured data
eu001UserData64Bytes	OUTPUT	ST_EKS2_Structure_Input_Eu001UserData64Bytes	User Data 64 Bytes structured data
eu001UserData90Bytes	OUTPUT	ST_EKS2_Structure_Input_Eu001UserData90Bytes	User Data 90 Bytes structured data
eu001MoFromEks2	OUTPUT	ST_EKS2_Structure_Input_Eu001Mo	MO structured data

Table 2: Parameters for the block

5.2.2. Calling the FB_EKS2_Eu001Base block

Call the *FB_EKS2_Eu001Base* function block and parameterize it in accordance with the application. Use a separate block instance for each EKS2 device.

The library provides the prepared *ST_EKS2_Eu001Integration* structure. The structure contains the inputs and outputs already including the associated data types. Using the structure is recommended.

```
FB_EKS2
1  FUNCTION_BLOCK FB_EKS2
2  VAR_INPUT
3  END_VAR
4  VAR_OUTPUT
5  END_VAR
6  VAR
7      fbEks2BaseExample   :   FB_EKS2_Eu001Base;
8      stEks2Base          :   ST_EKS2_Eu001Integration;
```

Fig. 18: Block interface

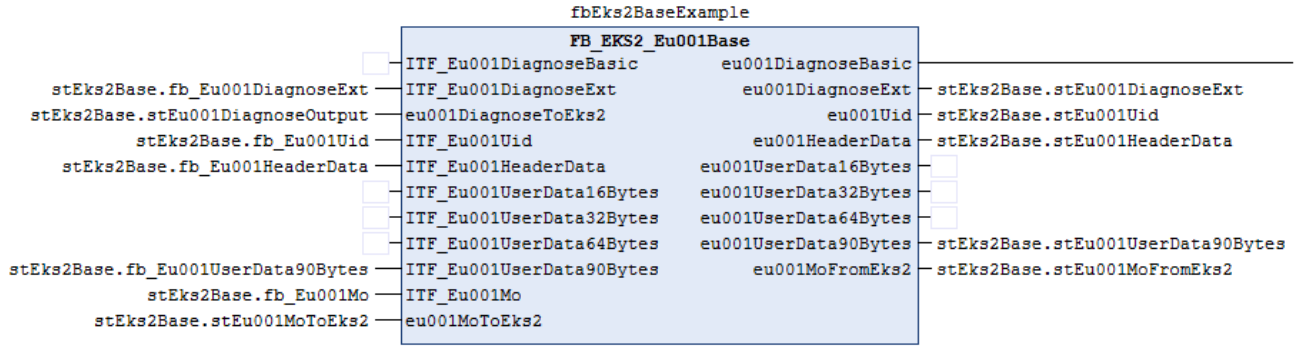


Fig. 19: Configured function block

5.2.3. Binding of the input and output areas

The first compilation process produces the process data that can be bound. I/O mapping binds these process data to the configured inputs and outputs of the EKS2 hardware.

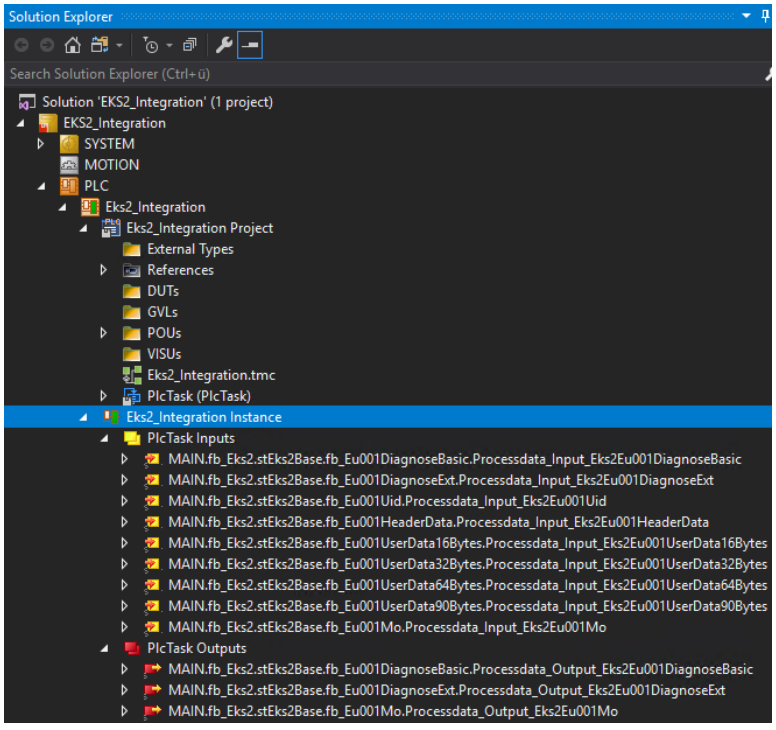


Fig. 20: Bindings produced

EN

The following application uses the hardware modules configured below. Deviations in the subterm numbering result from the respective configuration.

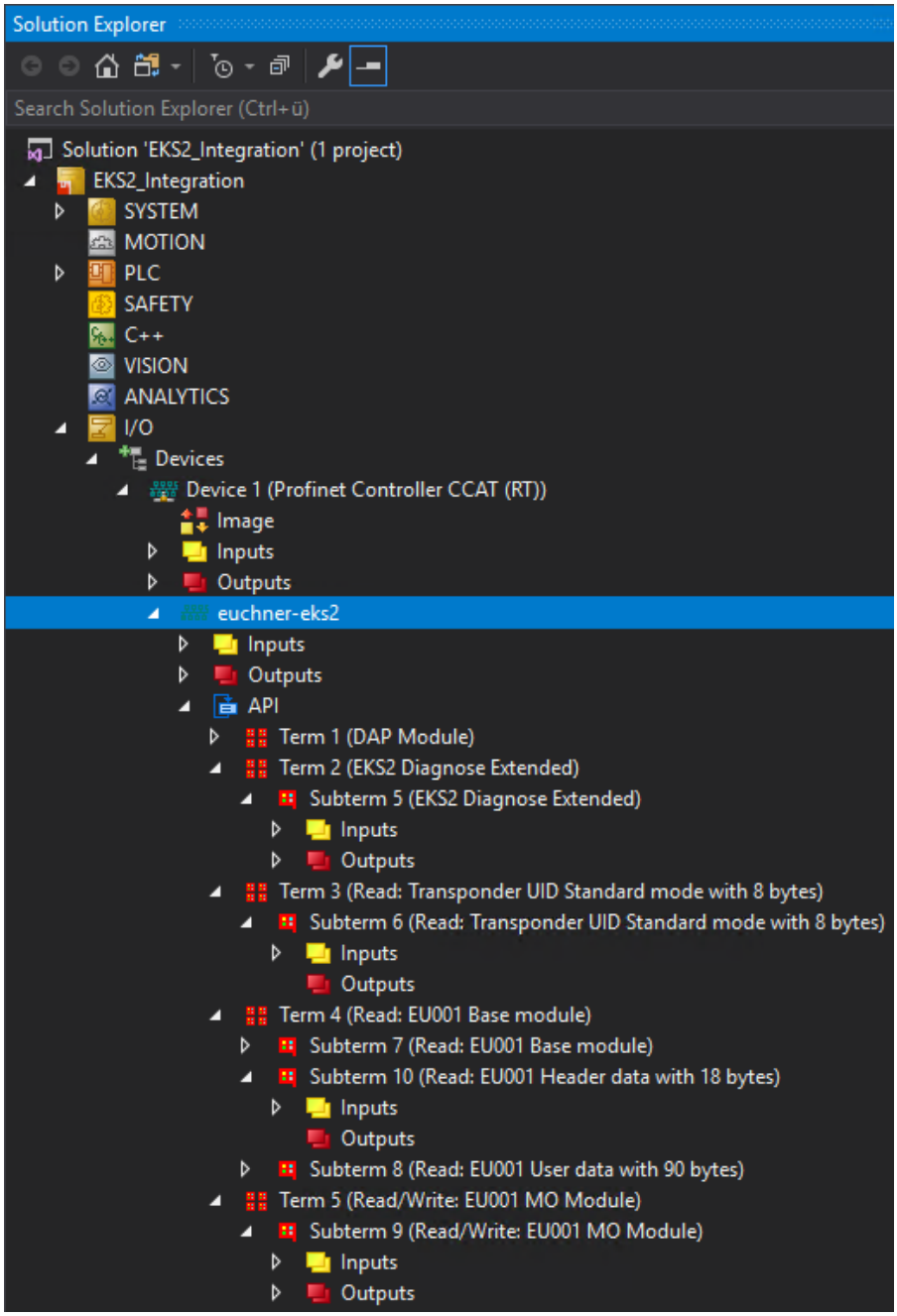


Fig. 21: Configured hardware

Process data	Use	Configured hardware
MAIN.fb_Eks2.stEks2Base.fb_Eu001DiagnoseExt.Processdata_Input_Eks2Eu001DiagnoseExt	INPUT	Subterm 5 (EKS2 Diagnose Extended)
MAIN.fb_Eks2.stEks2Base.fb_Eu001Uid.Processdata_Input_Eks2Eu001Uid	INPUT	Subterm 6 (Read: Transponder UID Standard mode with 8 bytes)
MAIN.fb_Eks2.stEks2Base.fb_Eu001HeaderData.Processdata_Input_Eks2Eu001HeaderData	INPUT	Subterm 10 (Read: EU001 Header data with 18 bytes)
MAIN.fb_Eks2.stEks2Base.fb_Eu001UserData90Bytes.Processdata_Input_Eks2Eu001UserData90Bytes	INPUT	Subterm 8 (Read: EU001 User data with 90 bytes)
MAIN.fb_Eks2.stEks2Base.fb_Eu001Mo.Processdata_Input_Eks2Eu001Mo	INPUT	Subterm 9 (Read/Write: EU001 MO Module)
MAIN.fb_Eks2.stEks2Base.fb_Eu001DiagnoseExt.Processdata_Output_Eks2Eu001DiagnoseExt	OUTPUT	Subterm 5 (EKS2 Diagnose Extended)
MAIN.fb_Eks2.stEks2Base.fb_Eu001Mo.Processdata_Output_Eks2Eu001Mo	OUTPUT	Subterm 9 (Read/Write: EU001 MO Module)

Table 3: I/O mapping between input and output process data and configured EKS2 hardware

The following example shows I/O mapping for Diagnose Extended.

Open the process data input (PDI) *MAIN...Eks2Eu001DiagnoseExt*, see *Fig. 20*. Then call the *Linked to...* function.

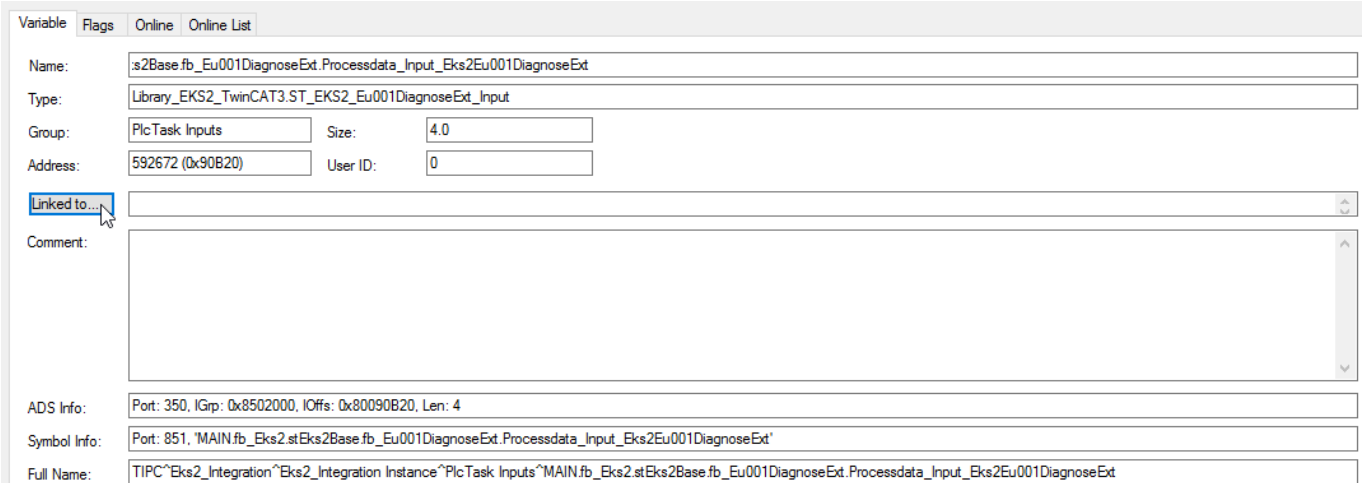


Fig. 22: I/O mapping

Another window then opens.

Select the corresponding EKS2 input for the *EKS2 Diagnose Extended* module there.

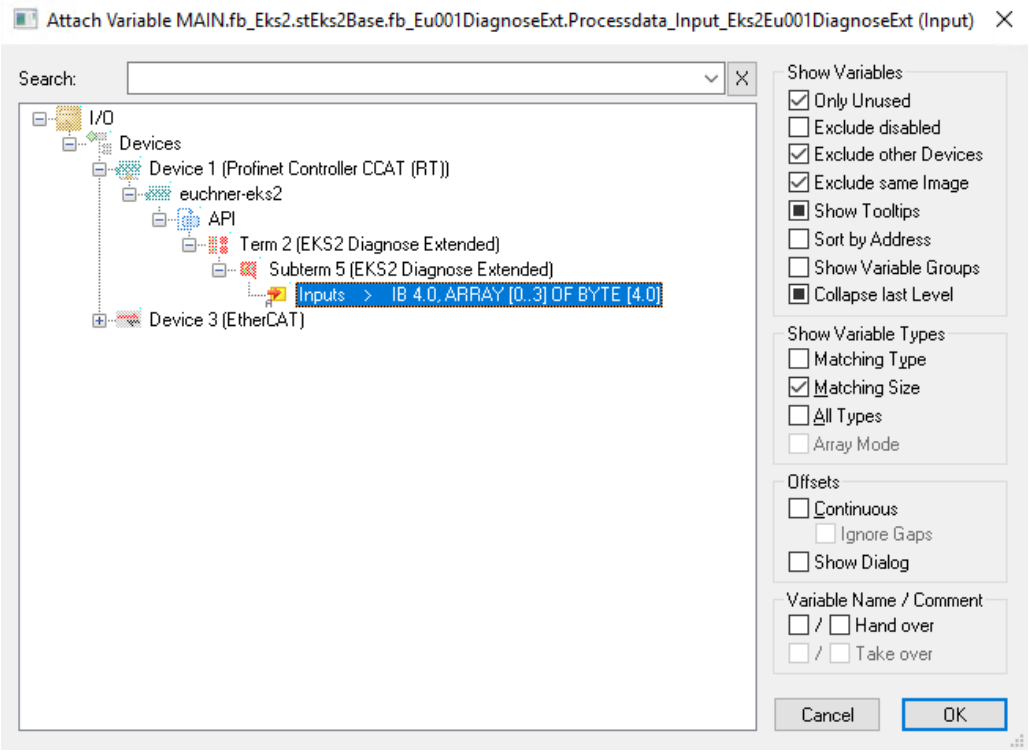


Fig. 23: Selection of the configured EKS2 input module

Perform I/O mapping for all input and output process data in accordance with the configured hardware (*Table 3*).

EN

5.2.4. EKS2 online data

After activating the configuration, establish an online connection to the control system and open the block interface. The figure below shows the prepared data based on the example of the advanced diagnostics data and the Electronic-Key's UID.

FB_EKS2 [Online]

EKS2_Integration.Eks2_Integration.MAIN.fb_Eks2

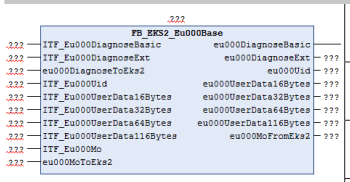
Expression	Type	Value	Prepared value	Address	Comment
fbEks2BaseExample	FB_EKS2_Eu001Base				
stEks2Base	ST_EKS2_Eu001Integration				
fb_Eu001DiagnoseBasic	FB_EKS2_Eu001DiagnoseBasic				Input: Function block associated with the API
fb_Eu001DiagnoseExt	FB_EKS2_Eu001DiagnoseExt				Input: Function block associated with the API
fb_Eu001Uid	FB_EKS2_Eu001Uid				Input: Function block associated with the API
fb_Eu001HeaderData	FB_EKS2_Eu001HeaderData				Input: Function block associated with the API
fb_Eu001UserData16Bytes	FB_EKS2_Eu001UserData16Bytes				Input: Function block associated with the API
fb_Eu001UserData32Bytes	FB_EKS2_Eu001UserData32Bytes				Input: Function block associated with the API
fb_Eu001UserData64Bytes	FB_EKS2_Eu001UserData64Bytes				Input: Function block associated with the API
fb_Eu001UserData90Bytes	FB_EKS2_Eu001UserData90Bytes				Input: Function block associated with the API
fb_Eu001Mo	FB_EKS2_Eu001Mo				Input: Function block associated with the API
stEu001DiagnoseOutput	ST_EKS2_Structure_Output_Eu001Diagnose				Input: Data to be written to EKS2
stEu001MoToEks2	ST_EKS2_Structure_Output_Eu001Mo				Input: Data to be written to EKS2
stEu001DiagnoseBasic	ST_EKS2_Structure_Input_Eu001DiagnoseBasic				Output: Available data
stEu001DiagnoseExt	ST_EKS2_Structure_Input_Eu001DiagnoseExt				Output: Available data
deviceReadyForOperation	BIT	TRUE			
electronicKeyDetected	BIT	TRUE			
acyclicalJobError	BIT	FALSE			
readyForAcyclicalJob	BIT	TRUE			
deviceError	BIT	FALSE			
cyclicalJobError	BIT	FALSE			
readyForCyclicalJob	BIT	TRUE			
cyclicalJobInProgress	BIT	FALSE			
noUserAreaIncluded	BIT	FALSE			
selectionOfSafeOperatingModeError	BIT	FALSE			
nc_1	BIT	FALSE			
nc_2	BIT	FALSE			
nc_3	BIT	FALSE			
electronicKeyInvalid	BIT	FALSE			
electronicKeyValid	BIT	TRUE			
nc_4	BIT	FALSE			
extDiagnosticByte	WORD	0			
stEu001Uid	ST_EKS2_Structure_Input_Eu001Uid				Output: Available data
uniqueId	ARRAY [0..7] OF BYTE				
uniqueId[0]	BYTE	16#04			
uniqueId[1]	BYTE	16#1F			
uniqueId[2]	BYTE	16#39			
uniqueId[3]	BYTE	16#2A			
uniqueId[4]	BYTE	16#C7			
uniqueId[5]	BYTE	16#10			
uniqueId[6]	BYTE	16#90			
uniqueId[7]	BYTE	16#00			

Fig. 24: EKS2 online data

5.3. EU000 data structure

5.3.1. Description of the block interface

The following table describes the parameters of the *FB_EKS2_Eu000Base* block.



Variable	Use	Data type	Description
ITF_Eu000DiagnoseBasic	INPUT	ITF_EKS2_Eu000DiagnoseBasic	Interface for Diagnose Basic module
ITF_Eu000DiagnoseExt	INPUT	ITF_EKS2_Eu000DiagnoseExt	Interface for Diagnose Extended module
eu000DiagnoseToEks2	INPUT	ST_EKS2_Structure_Output_Eu000Diagnose	Output data for Diagnose (Base and Extended) module
ITF_Eu000Uid	INPUT	ITF_EKS2_Eu000Uid	Interface for UID module
ITF_Eu000UserData16Bytes	INPUT	ITF_EKS2_Eu000UserData16Bytes	Interface for User Data 16 Bytes module
ITF_Eu000UserData32Bytes	INPUT	ITF_EKS2_Eu000UserData32Bytes	Interface for User Data 32 Bytes module
ITF_Eu000UserData64Bytes	INPUT	ITF_EKS2_Eu000UserData64Bytes	Interface for User Data 64 Bytes module
ITF_Eu000UserData116Bytes	INPUT	ITF_EKS2_Eu000UserData116Bytes	Interface for User Data 116 Bytes module
ITF_Eu000Mo	INPUT	ITF_EKS2_Eu000Mo	Interface for MO module
eu000MoToEks2	INPUT	ST_EKS2_Structure_Output_Eu000Mo	Output data for MO module
eu000DiagnoseBasic	OUTPUT	ST_EKS2_Structure_Input_Eu000DiagnoseBasic	Diagnose Basic structured data
eu000DiagnoseExt	OUTPUT	ST_EKS2_Structure_Input_Eu000DiagnoseExt	Diagnose Extended structured data
eu000Uid	OUTPUT	ST_EKS2_Structure_Input_Eu000Uid	UID structured data
eu000UserData16Bytes	OUTPUT	ST_EKS2_Structure_Input_Eu000UserData16Bytes	User Data 16 Bytes structured data
eu000UserData32Bytes	OUTPUT	ST_EKS2_Structure_Input_Eu000UserData32Bytes	User Data 32 Bytes structured data
eu000UserData64Bytes	OUTPUT	ST_EKS2_Structure_Input_Eu000UserData64Bytes	User Data 64 Bytes structured data
eu000UserData116Bytes	OUTPUT	ST_EKS2_Structure_Input_Eu000UserData116Bytes	User Data 116 Bytes structured data
eu000MoFromEks2	OUTPUT	ST_EKS2_Structure_Input_Eu000Mo	MO structured data

Table 4: Parameters for the block

EN

5.3.2. Calling the FB_EKS2_Eu000Base block

Call the *FB_EKS2_Eu000Base* function block and parameterize it in accordance with the application. Use a separate block instance for each EKS2 device.

The library provides the prepared *ST_EKS2_Eu000Integration* structure. The structure contains the inputs and outputs already including the associated data types. Using the structure is recommended.

```
FB_EKS2
2   VAR_INPUT
3   END_VAR
4   VAR_OUTPUT
5   END_VAR
6   VAR
7       fbEks2BaseExample    :   FB_EKS2_Eu000Base;
8       stEks2Base           :   ST_EKS2_Eu000Integration;
```

Fig. 25: Block interface

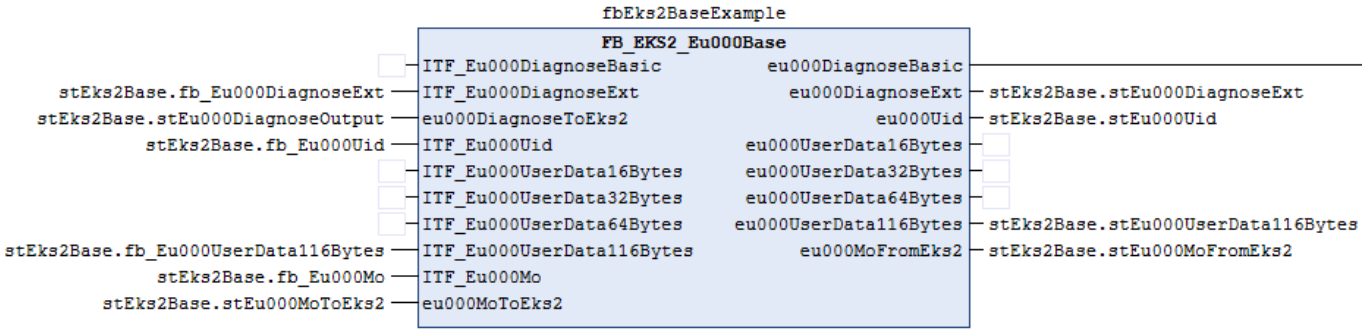


Fig. 26: Configured function block

5.3.3. Binding of the input and output areas

The first compilation process produces the process data that can be bound. I/O mapping binds these process data to the configured inputs and outputs of the EKS2 hardware.

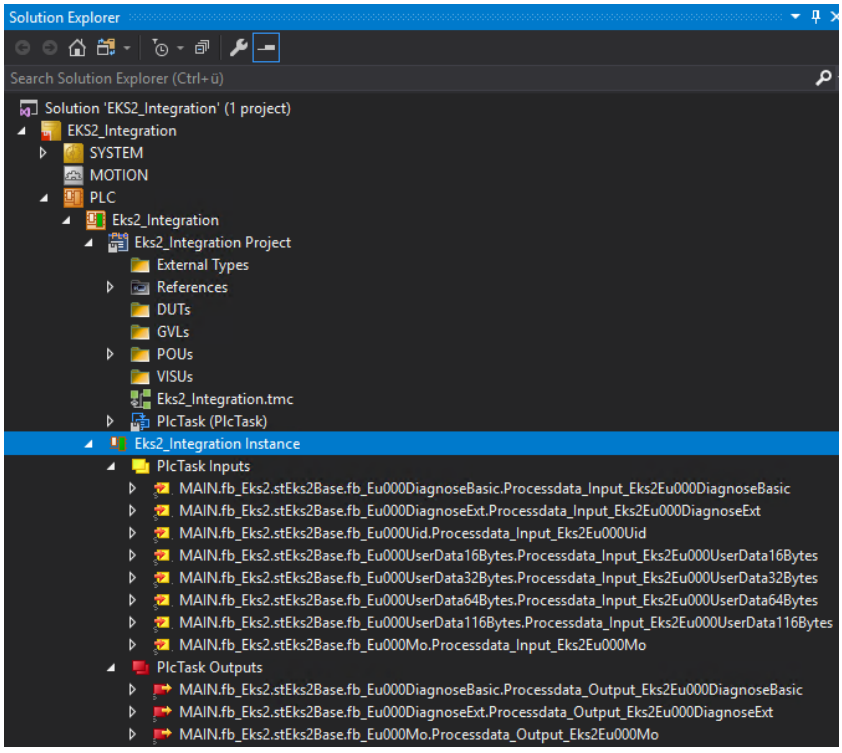


Fig. 27: Bindings produced

The following application uses the hardware modules configured below. Deviations in the subterm numbering result from the respective configuration.

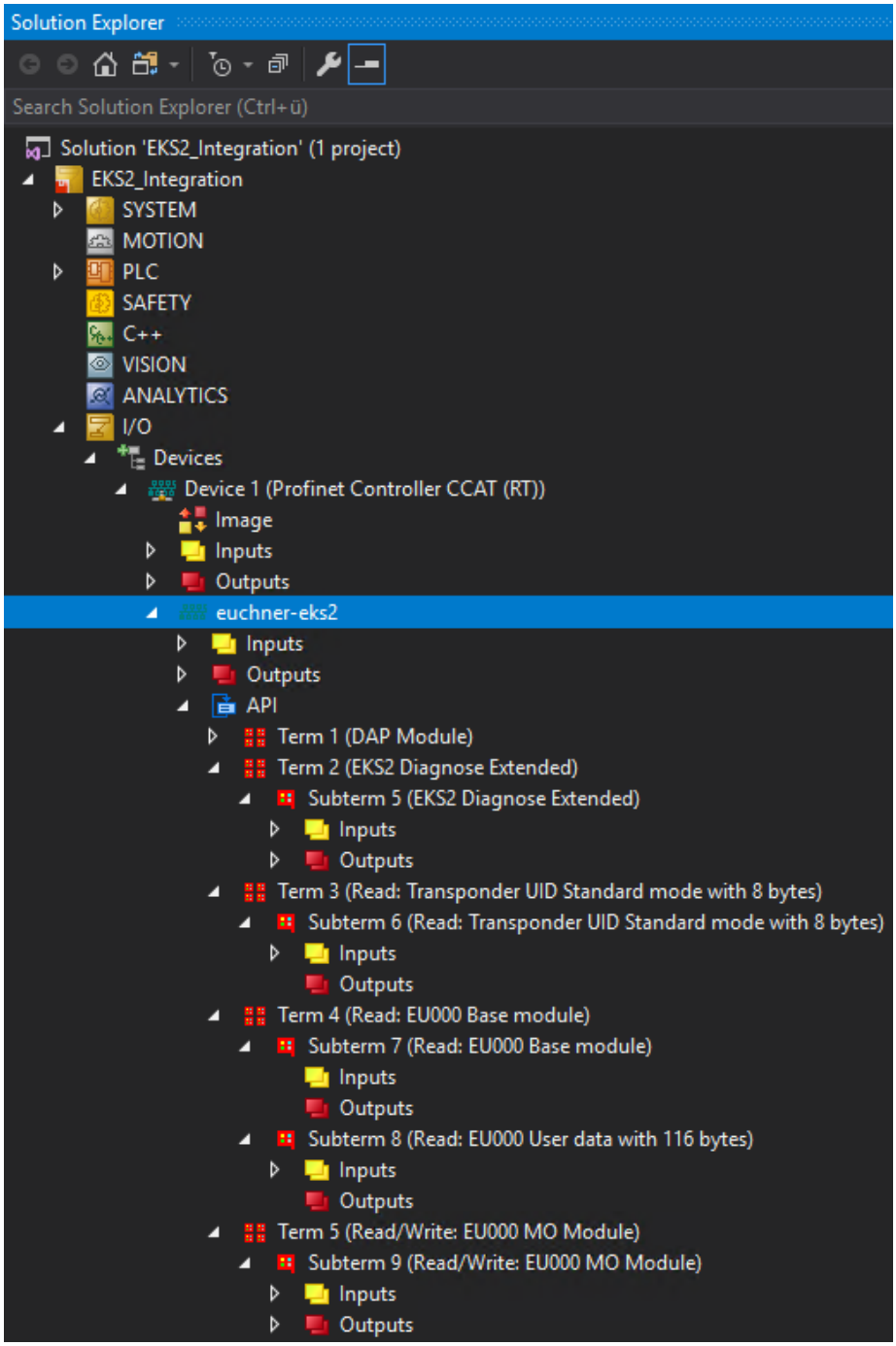


Fig. 28: Configured hardware

Process data	Use	Configured hardware
MAIN.fb_Eks2.stEks2Base.fb_Eu000DiagnoseExt.Processdata_Input_Eks2Eu000DiagnoseExt	INPUT	Subterm 5 (EKS2 Diagnose Extended)
MAIN.fb_Eks2.stEks2Base.fb_Eu000Uid.Processdata_Input_Eks2Eu000Uid	INPUT	Subterm 6 (Read: Transponder UID Standard mode with 8 bytes)
MAIN.fb_Eks2.stEks2Base.fb_Eu000UserData116Bytes.Processdata_Input_Eks2Eu000UserData-116Bytes	INPUT	Subterm 8 (Read: EU000 User data with 116 bytes)
MAIN.fb_Eks2.stEks2Base.fb_Eu000Mo.Processdata_Input_Eks2Eu000Mo	INPUT	Subterm 9 (Read/Write: EU000 MO Module)
MAIN.fb_Eks2.stEks2Base.fb_Eu000DiagnoseExt.Processdata_Output_Eks2Eu000DiagnoseExt	OUTPUT	Subterm 5 (EKS2 Diagnose Extended)
MAIN.fb_Eks2.stEks2Base.fb_Eu000Mo.Processdata_Output_Eks2Eu000Mo	OUTPUT	Subterm 9 (Read/Write: EU000 MO Module)

Table 5: I/O mapping between input and output process data and configured EKS2 hardware

EN

The following example shows I/O mapping for Diagnose Extended.

Open the process data input (PDI) *MAIN...Eks2Eu000DiagnoseExt*, see *Fig. 27*. Then call the *Linked to...* function.

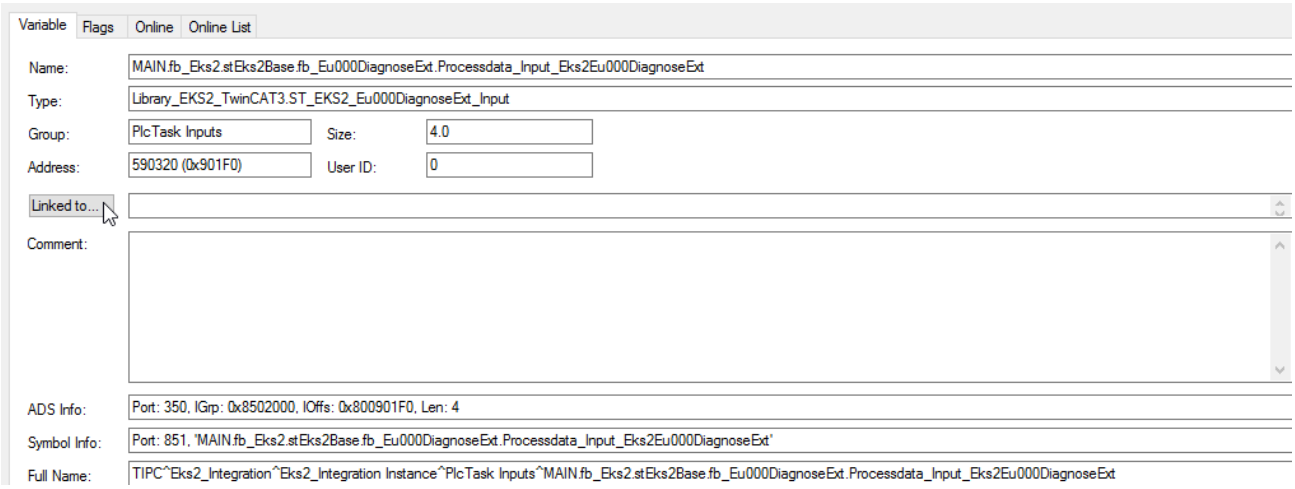


Fig. 29: I/O mapping

Another window then opens.

Select the corresponding EKS2 input for the *EKS2 Diagnose Extended* module there.

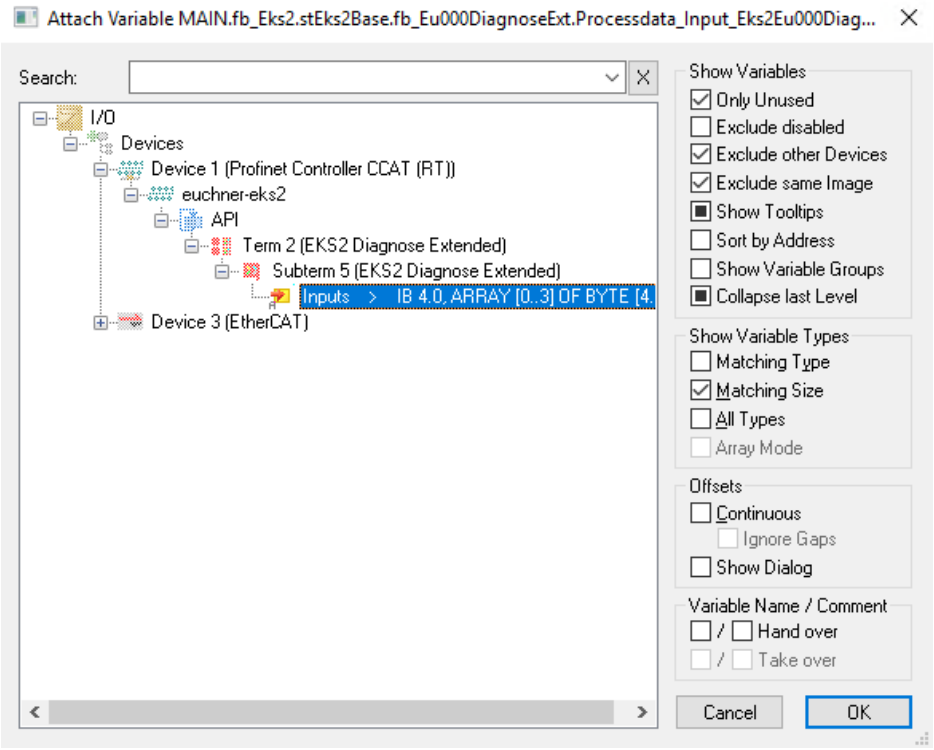


Fig. 30: Selection of the configured EKS2 input module

Perform I/O mapping for all input and output process data corresponding to the configured hardware (*Table 5*).

5.3.4. EKS2 online data

After activating the configuration, establish an online connection to the control system and open the block interface. The figure below shows the prepared data based on the example of the advanced diagnostics data and the Electronic-Key's UID.

FB_EKS2 [Online]

EKS2_Integration.Eks2_Integration.MAIN.fb_Eks2

Expression	Type	Value	Prepared value	Address	Comment
fbEks2BaseExample	FB_EKS2_Eu000Base				
stEks2Base	ST_EKS2_Eu000Integration				
fb_Eu000DiagnoseBasic	FB_EKS2_Eu000DiagnoseBasic				Input: Function block associated with the API
fb_Eu000DiagnoseExt	FB_EKS2_Eu000DiagnoseExt				Input: Function block associated with the API
fb_Eu000Uid	FB_EKS2_Eu000Uid				Input: Function block associated with the API
fb_Eu000UserData16Bytes	FB_EKS2_Eu000UserData16Bytes				Input: Function block associated with the API
fb_Eu000UserData32Bytes	FB_EKS2_Eu000UserData32Bytes				Input: Function block associated with the API
fb_Eu000UserData64Bytes	FB_EKS2_Eu000UserData64Bytes				Input: Function block associated with the API
fb_Eu000UserData116Bytes	FB_EKS2_Eu000UserData116Bytes				Input: Function block associated with the API
fb_Eu000Mo	FB_EKS2_Eu000Mo				Input: Function block associated with the API
stEu000DiagnoseOutput	ST_EKS2_Structure_Output_Eu000Diagnose				Input: Data to be written to EKS2
stEu000MoToEks2	ST_EKS2_Structure_Output_Eu000Mo				Input: Data to be written to EKS2
stEu000DiagnoseBasic	ST_EKS2_Structure_Input_Eu000DiagnoseBasic				Output: Available data
stEu000DiagnoseExt	ST_EKS2_Structure_Input_Eu000DiagnoseExt				Output: Available data
deviceReadyForOperation	BIT	TRUE			
electronicKeyDetected	BIT	TRUE			
acyclicalJobError	BIT	FALSE			
readyForAcyclicalJob	BIT	TRUE			
deviceError	BIT	FALSE			
cyclicalJobError	BIT	FALSE			
readyForCyclicalJob	BIT	TRUE			
cyclicalJobInProgress	BIT	FALSE			
noUserAreaIncluded	BIT	FALSE			
selectionOfSafeOperatingModeError	BIT	FALSE			
nc_1	BIT	FALSE			
nc_2	BIT	FALSE			
nc_3	BIT	FALSE			
electronicKeyInvalid	BIT	FALSE			
electronicKeyValid	BIT	TRUE			
nc_4	BIT	FALSE			
extDiagnosticByte	ARRAY [0..1] OF BYTE				
stEu000Uid	ST_EKS2_Structure_Input_Eu000Uid				Output: Available data
uniqueId	ARRAY [0..7] OF BYTE				
uniqueId[0]	BYTE	16#04			
uniqueId[1]	BYTE	16#1E			
uniqueId[2]	BYTE	16#2F			
uniqueId[3]	BYTE	16#2A			
uniqueId[4]	BYTE	16#C7			
uniqueId[5]	BYTE	16#10			
uniqueId[6]	BYTE	16#90			

Fig. 31: EKS2 online data

EN

6. Important notice – please observe carefully!

This document is intended for a design engineer who possesses the requisite knowledge in safety engineering and knows the applicable standards, e.g. through training for qualification as a safety engineer. Only with the appropriate qualification is it possible to integrate the example provided into a complete safety chain.

The example represents only part of a complete safety chain and does not fulfill any safety function on its own. In order to fulfill a safety function, the energy switch-off function for the danger zone and the software must also be considered in the safety evaluation, for example.

The applications provided are only examples for solving certain safety tasks for protecting safety doors. The examples cannot be comprehensive due to the application-dependent and individual protection goals within a machine/installation.

If questions concerning this example remain open, please contact us directly.

According to the Machinery Directive 2006/42/EC, the design engineer of a machine or installation has the obligation to perform a risk assessment and take measures to reduce the risk. While doing this, the engineer must comply with the applicable national and international safety standards. Standards generally represent the current state-of-the-art. Therefore, the design engineer should continuously inform himself about changes in the standards and adapt his considerations to them. Relevant standards for functional safety include EN ISO 13849 and EN 62061. This application must be regarded only as assistance for the considerations about safety measures.

The design engineer of a machine/installation has the obligation to assess the safety engineering himself. The examples must not be used for an assessment, because only a small excerpt of a complete safety function was considered in terms of safety engineering here.

In order to be able to use the safety switch applications correctly on safety doors, it is indispensable to observe the standards EN ISO 13849-1, EN ISO 14119 and all relevant C-standards for the respective machine type. Under no circumstances does this document replace the engineer's own risk assessment, and it cannot serve as the basis for a fault assessment.

In particular in relation to a fault exclusion, it must be noted that a fault can be excluded only by the machine's or installation's design engineer and this action requires justification. A general fault exclusion is not possible. More information about fault exclusion can be found in EN ISO 13849-2.

Changes to products or within assemblies from third-party suppliers used in this example can lead to the function no longer being ensured or the safety assessment having to be adapted. In any event, the information in the operating instructions on the part of EUCHNER, as well as on the part of third-party suppliers, must be used as the basis before this application is integrated into an overall safety function. If contradictions should arise between the operating instructions and this document, please contact us directly.

Use of brand names and company names

All brand names and company names stated are the property of the related manufacturer. They are used only for the clear identification of compatible peripheral devices and operating environments in relation to our products.

EUCHNER GmbH + Co. KG
Kohlhammerstraße 16
70771 Leinfelden-Echterdingen
info@euchner.de
www.euchner.com

Edition:
AP000287-01-05/26
Title:
Application EKS2
EKS2 – Integration of EKS2 with PROFINET Interface in BECKHOFF TwinCAT 3