

Application



Integration of EKS with TCP/IP Interface in Rockwell Studio 5000®

Contents

1.	Abou	ut this document	
	1.1.	Version	3
	1.2.	Scope	3
	1.3.	Target group	3
	1.4.	Supplementary documents	3
	1.5.	Notice	3
2.	Com	ponents/modules used	4
	2.1.	EUCHNER	4
	2.2.	Others	4
	2.3.	Software	4
3.	Fund	ctional description	4
4.	Impo	orting the Add-On Instruction	5
5.	Integ	grating the Add-On Instruction	7
	5.1.	Variable table for the Add-On Instruction	7
	5.2.	Creating the controller tags	8
	5.3.	Configuring the messages	
	5.4.	Activating socket connection	16
	5.5.	Command for writing Electronic-Key	17
	5.6.	Configuring the parameters for the IP address	
	5.7.	Starting the program	
	5.8.	Writing the Electronic-Key	20
6.	Impo	ortant note – please observe carefully!	

1. About this document

1.1. Version

Version	Date	Change/addition	Chapter
01-02/20	1/10/2020	Prepared	All
02-08/20	8/3/2020	Addition: writing Electronic-Key	5.5/5.8
02.04/21	4/16/2021	- Data type of "JobFinishedActiveTime" variable changed from UINT to INT	5.1
03-04/21	4/10/2021	- "IMPORTANT!" added: network card selection	5.3

1.2. Scope

The purpose of this document is the integration and configuration of the EKS with TCP/IP interface in Rockwell Studio 5000 Logix Designer[®].

1.3. Target group

Design engineers and installation planners for safety systems on machines, as well as setup and servicing staff possessing special expertise in handling safety components as well as expertise in the installation, setup, programming and diagnostics of programmable logic controllers (PLCs) and bus systems.

1.4. Supplementary documents

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

Document title (document number)	Contents	
Manual (2100420)	Electronic-Key-System Manual for Electronic-Key adapter EKS and EKS FSA with Ethernet TCP/IP interface	www
Possibly enclosed data sheets	Item-specific information about deviations or additions	

1.5. Notice

This application is based on the manual for the EKS with TCP/IP interface. Please refer to the manual for the technical details and other information. In the rest of this document, the EKS with TCP/IP interface is referred to as the "EKS" for short.

2. Components/modules used

2.1. EUCHNER

 (\mathbf{i})

Description	Order number / item
EKS with TCP/IP interface	100401 / EKS-A-IEX-G01-ST02/03
	099265 / EKS-A-IEXA-G01-ST02/03/04

TIP!

More information and downloads about the aforementioned EUCHNER products can be found at <u>www.euchner.com</u>. Simply enter the order number in the search box.

2.2. Others

Description	Order number / item
1756-L81ES GuardLogix® 5580 Safety Controller	1756-L81ES
1756-L8SP GuardLogix® 5580 Safety Partner	1756-L8SP

2.3. Software

Description	Version
Studio 5000 Logix Designer	Version 32.01.00 - Professional Edition

3. Functional description

EKS TCP/IP devices are read/write systems with electronics for the inductive bidirectional interface to the transponder and interface electronics.

The system is connected via the integrated TCP/IP interface, which is designed as an RJ45 socket. A separate switch may be required for the TCP/IP connection. The EKS does not have an integrated switch.

The current state of the Electronic-Key adapter is displayed using a 3-color LED.

The Electronic-Key is placed on the Electronic-Key adapter for operation. The power supply for the transponder and the data are transferred between the Electronic-Key adapter and the Electronic-Key without using any contacts.

The data transmission between the control system and EKS is realized using an Add-On Instruction (AOI). The AOI handles establishing communication between the control system and EKS as well as sending and receiving the TCP/IP communication telegrams.

The AOI can be downloaded from <u>www.euchner.com</u> in the area Service/Downloads/Software/Sample files and Libraries/EKS.

4. Importing the Add-On Instruction

1. By right-clicking Add-On Instructions in the Controller Organizer, open the context menu and select Import Add-On Instruction.



- Fig. 1: Add-On Instruction context menu
- 2. Select the folder with the downloaded AOI and open the AOI using Open.



Fig. 2: Import Add-On Instruction dialog box

3. Complete the import in the Import Configuration window using OK without making any changes.

ℤ ⅔ Find: Find: Find: Find/Replace
Find Within: Final Name
Import Content:
Add-On Instructions Configure Add-On Instruction Properties
ADI_EKS_TCPIP_V32_YYYYM Import Name: AOI_EKS_TCPIP_V32_YYYYMMDD
- Routines Operation: Use Existing
Geferences Geferences (i) References will be imported as Configured in the References folders
-Lo Errors/Warnings* Final Name: A01_EKS_TCPIP_V32_YYYMM V Collision Details
Description:
Class: Standard
Revision: v1.0
Revision Note:
Vendor:
OK Cancel Help
Ready

Fig. 3: Import Configuration window

The following data types and the Add-On Instruction are added to the Controller Organizer during the import.



Fig. 4: Imported data types and AOI

EUCHNER

5. Integrating the Add-On Instruction

5.1. Variable table for the Add-On Instruction

		-
AOI_EKS_TOPIP_V32_YYYYMMI	00	
AOI_EKS_TCPIP_V32_YYY		1
SktConExable	77	CSktConToServ)-
SktEKSData	2	
SktTimeout	?	CSktUsed)-
	77	
SktCreateMsg	2	SktError >
SktConnectMsg	2	
SktRaadMsg	2	CSatOpen -
SktWriteMag	2	
SktDelMsg	2	CSktRead >
EKSReadMode	?	
	22	C3ktWiteMan
lobFinishedActiveTime	72	
EKSStartAddressRead	2	CEKSKevIN)
	77	
KSNumberOfDytesRead	2	CEKSStatusMessags>
	77	
KSKeyDataRead	2	ClobFinished >
KSStartAddressWrite	2	
	77	
3KSNumberOfBytesWrite	?	
	77	
EKSKeyDatsWrite	2	
EKSStatusNumber	72	

Variable	Use	Data type	Description
AOI_EKS_TCPIP_V32_YYYYMMDD	-	AOI_EKS_TCPIP_ V32_YYYYMMDD	Instance for the AOI
SktConEnable	Input	BOOL	Activates TCP/IP socket connection
SktEKSData	InOut	typeSktEKSData	This variable contains all necessary data for the messages
SktTimeout	Input	DINT	Time limit for the messages (in ms)
SktCreateMsg	InOut	MESSAGE	Message for creating the TCP/IP socket
SktConnectMsg	InOut	MESSAGE	Message for establishing the TCP/IP socket con- nection
SktReadMsg	InOut	MESSAGE	Message for reading the content of the TCP/IP telegram
SktWriteMsg	InOut	MESSAGE	Message for writing the TCP/IP telegram
SktDelMsg	InOut	MESSAGE	Message for deleting the TCP/IP socket connection
EKSReadMode	Input	SINT	Mode for the request 1= manual; 2= automatic
JobFinishedActiveTime	Input	INT	Time value (in ms) indicating how long the JobFin- ished bit is to remain active after the write process
EKSStartAdressRead	Input	SINT	Start address for the Electronic-Key data to be requested
EKSNumberOfBytesRead	Input	SINT	Amount of Electronic-Key data to be requested
EKSKeyDataRead	InOut	SINT[124]	Reply with the user data from the EKS Electronic-Key
EKSStartAddressWrite	Input	SINT	Start address of the Electronic-Key data to be written
EKSNumberOfBytesWrite	Input	SINT	Amount of Electronic-Key data to be written
EKSKeyDataWrite	InOut	SINT[116]	Electronic-Key data to be written
EKSStatusNumber	Output	INT	EKS status
SktConToServ	Output	BOOL	Attempting to establish a server connection
SktUsed	Output	BOOL	TRUE = the socket connection process is started; FALSE = the socket connection process is stopped
SktError	Output	BOOL	Socket connection error
SktOpen	Output	BOOL	Socket connection open
SktRead	Output	BOOL	Receiving telegram
SktWrite	Output	BOOL	Sending telegram
EKSKeylN	Output	BOOL	EKS Electronic-Key placed in the Electronic-Key adapter
EKSStatusMessage	Output	BOOL	An EKS status message has been received
JobFinished	Output	BOOL	Write process completed

Table 1: AOI variable table

5.2. Creating the controller tags

1. Open a program (e.g. MainRoutine) and drag the AOI to a new rung using drag & drop.



Fig. 5: Adding the AOI to the main program

2. Create the instance of the AOI. Type the name of the instance in the field for the variable AOI_EKS_TCPIP_V32_YYYMMDD and open the context menu by right-clicking. In this example, select New "AOI_EKS_Milling."





3. The necessary information is already entered in the New Tag window. Click Create to create the variable.

New Tag		×
<u>N</u> ame:	AOI_EKS_Milling	Create 🗸 🗸
Description:	^	Cancel
		Help
	~	
<u>U</u> sage:	<controller></controller>	
Typ <u>e</u> :	Base ~ <u>C</u> onnection	
Alias <u>F</u> or:	~	
Data <u>T</u> ype:	AOI_EKS_TCPIP_V32_YYYYMMDE	
Para <u>m</u> eter Connection:	×	
<u>S</u> cope:	₽ AP000250	
Cl <u>a</u> ss:	Standard ~	
E <u>x</u> ternal Access:	Read/Write ~	
St <u>y</u> le:	\checkmark	
<u>C</u> onstant		
Seguencing		
Open Config	guration	
Open <u>P</u> aran	neter Connections	

Fig. 7: Creating instance variable

4. Create the variable for the EKS socket connection. Type the variable name in the related variable field and open the context menu by right-clicking. In this example, select *New "EKS_Milling.*" Then create the variable using *Create* without making any changes.



Fig. 8: Creating variable for the EKS socket data

ΕN

5. Create the variable for the Socket Create Message. Type the variable name in the related variable field and open the context menu by right-clicking. In this example, select *New "EKS_Skt_Create_Msg."*

💰 Logix Designer - AP000250 [1756-L81ES 32.12] – □ ×					
🗏 MainProgram - MainRoutine* 🗙		-			
€ , Q , ₩, ½, [¬] , [¬] , [™]					
0 ☎ AOI EKS TCPIP_V32_YYYYY AOI_EKS_TCPIP_V32_YYYY SktConEnable SktEKSData SktTimeout SktCreateMsg EKS SktConnectMsg SktReadMsg SktReadMsg SktWriteMsg	/MMDD AOI_EKS_Milling 0 ← EKS_Milling 2000 -(SktConToServ)- -(SktUsed)- KS Skt Create Msg New "EKS_Skt_Create_Msg" X Cut Instruction StratX	^			
SktDelMsg EKSReadMode	Copy Instruction Strg+C				
JobFinishedActiveTime	Paste Strg+V				
EKSStartAddressRead EKSNumberOfBytesRead	Delete Instruction Entf Add Ladder Element Alt+Einfg				
EKSKeyDataRead EKSStartAddressWrite	Edit Main Operand Description Strg+D				
EKSNumberOfBytesWrite	Save Instruction Defaults Clear Instruction Defaults				
EKSKeyDataWrite EKSStatusNumber	Remove Force				

Fig. 9: Creating the Socket Create Message variable

6. Then create the variable using *Create* without making any changes.

New Tag		×
<u>N</u> ame:	EKS_Skt_Create_Msg	Create 🗸 🔻
Description:	^	Cancel
		Help
	~	
<u>U</u> sage:	<controller></controller>	
Тур <u>е</u> :	Base ∨ <u>C</u> onnection	
Alias <u>F</u> or:	~	
Data <u>T</u> ype:	MESSAGE	
Para <u>m</u> eter Connection:	~	
<u>S</u> cope:	P000250	
Cl <u>a</u> ss:	Standard ~	
E <u>x</u> ternal Access:	Read/Write \vee	
St <u>y</u> le:	~	
<u>C</u> onstant		
Seguencing	9	
Open MES	SAGE Configuration	
Open <u>P</u> arar	meter Connections	

Fig. 10: Creating the Socket Create Message variable

\mathbf{i}	

IMPORTANT

Repeat the creation of the message variables (steps 5 and 6) for the variables *SktConnectMsg*, *SktReadMsg*, *SktWriteMsg* and *SktDelMsg*. Additionally create the *EKSKeyDataRead* and *EKSKeyDataWrite* variables.

7. Entering values for the input variables.

Variable	Value
SktTimeout	2,000 (ms)
EKSReadMode	 1 = manual mode; the Electronic-Key data are received by triggering the variable SktWrite 2 = automatic mode; the reception of the Electronic-Key data is triggered by reading the Electronic-Key status Key/N (see manual)
JobFinishedActiveTime	Variable (value from 0 to 65,535) → Time value (in ms) indicating how long the JobFinished bit is to remain active after the write process
EKSStartAddressRead	Variable (value from 0 to 116) \rightarrow Start address for the user data used
EKSNumberOfBytesRead	Variable (value from 0 to 124) \rightarrow Number of bytes to be read
EKSStartAddressWrite	Variable (value from 0 to 112) \rightarrow Start address for the user data used
EKSNumberOfBytesWrite	Variable (value from 4 to 116) \rightarrow Number of bytes to be written

 \mathbf{i}

IMPORTANT

On the Electronic-Key read/write with 116 bytes freely programmable, the memory is organized in 4-byte blocks. This means the start address for writing must be given in the range byte number 0 to byte number 112, always in 4-byte steps (byte number 0, 4, 8 ... 112). Also, a multiple of 4-byte-sized blocks must always be written (4, 8, 12 ... 116 bytes)!

However, during reading it is possible to access the memory byte-by-byte without the above-mentioned restriction for writing.

The Electronic-Key read/write also has a unique 8-byte serial number that is permanently written to the memory during the Electronic-Key production process. The serial number therefore cannot be changed. This serial number is used for secure differentiation of every single Electronic-Key. It is necessary that all 8 bytes are completely evaluated for secure differentiation. The serial number is appended to the freely programmable memory. The serial number can be read by entering the start address byte number 116 and the number of bytes 8.



5.3. Configuring the messages

1. Open the Configuration Dialog for SktCreateMsg.

Fig. 12: Opening the Configuration Dialog

2. Select Service Type: Socket Create. Configure the following parameters:

Parameter	Value
Source Element	EKS_Milling.CreateSrc
Destination Element	EKS_Milling.Inst
Path	THIS (you will find the Path parameter on the Communication tab)

IMPORTANT

The network interface via which the EKS is to communicate must be specified in the *Path* parameter. If there are several network cards, make sure that the correct network card is selected in the *Path* parameter. This concerns every *Message* configuration.

Message Configuration - EKS_Skt_Create_Msg	Message Configuration - EKS_Skt_Create_Msg
Configuration* Communication Tag	Configuration* Communication* Tag
Message Type: CIP Generic Service Socket Create Type: Source Length: Service 4b (Hex) Class: 342 (Hex) Instance: 0 Attribute: 0 (Hex) Element: EKS_Milling.CreateSrc New Tag	Path: THIS THIS THIS THIS This Broadcast: Communication Method
◯ Enable ◯ Enable Waiting ◯ Start ◯ Done Done Length: 0	Cenable Cenable Waiting Start ODone Done Length: 0
Cerror Code: Extended Error Code: Timed Out Timed Out Timed Out Timed Out Timed Out	○ Error Code: Extended Error Code: ☐ Timed Out ♥ Error Path: Error Text:
OK Abbrechen Obernehmen Hilfe	OK Abbrechen Obemehmen Hilfe

Fig. 13: Configuring parameters for *Socket Create Message*

3. Open the Configuration Dialog for SktConnectMsg. Select Service Type: OpenConnection. Configure the following parameters:

Parameter	Value
Source Element	EKS_Milling.OpenSrc
Source Length	1
Instance	0
Path	THIS (you will find the Path parameter on the Communication tab)

Message Configuration - EKS_Skt_Connect_Msg	Message Configuration
Message Configuration - EKS_Skt_Connect_Msg Configuration* Service Code: Service Code: Code: O Attribute: O Hexx) Element: New Tag	Message Configuration Configuration Configuration Communication THIS Broadcast: Communication Method CIP DH+ Channel: Communication Method CIP DH+ Channel: Communication Method CIP CIP CIP CIP CIP CIP CIP CIP CIP CI
O Enable ○ Enable Waiting ○ Start ○ Done Done Length: 0 O Error Code: Extended Error Code: □ Timed Out ◆ Error Path: Error Text: OK Abbrechen Übernehmen Hilfe	C Enable ○ Enable Waiting ○ Start ○ Done Done Length: 0 Error Code: Extended Error Code: □ Timed Out ◆ Error Path: Error Text: OK Abbrechen Ubemehmen Hilfe

Fig. 14: Configuring parameters for OpenConnection Message

4. Open the Configuration Dialog for SktReadMsg. Select Service Type: ReadSocket. Configure the following parameters:

Parameter	Value
Source Element	EKS_Milling.ReadSrc
Destination Element	EKS_Milling.ReadResponse
Instance	0
Path	THIS (you will find the Path parameter on the Communication tab).

Message Configuration - EKS_Skt_Read_Msg	Message Configuration
Configuration* Communication Tag	Configuration* Communication* Tag
Message Type: CIP Generic Service ReadSocket Type: Source Element: Service ad Code: ad Instance: 0 Attribute: 0 (Hex) Hex) Element: Element: Exs_Miling.ReadResponse New Tag	Path: THIS THIS Broadcast: Communication Method © CIP DH+ Channel: A* Destination Link: 0 O <po <p="">O <po <p="">O <po <="" <po="" description="" p=""></po></po></po>
Carable Carable Waiting Caraba Cone Done Length: 0	⊖ Enable ⊖ Enable Waiting ⊖ Start ⊖ Done Done Length: 0
O Error Code: Extended Error Code: Timed Out ♥ Error Path: Error Text:	○ Error Code: Extended Error Code:
OK Abbrechen Übernehmen Hilfe	OK Abbrechen Obernehmen Hilfe

Fig. 15: Configuring parameters for ReadSocket Message

5. Open the Configuration Dialog for SktWriteMsg. Select Service Type: WriteSocket. Configure the following parameters:

Parameter	Value
Source Element	EKS_Milling.WriteSrc
Destination Element	EKS_Milling.WriteSizeSent
Source Length	1
Instance	0
Path	THIS (you will find the Path parameter on the Communication tab)

Message Configuration - EKS_Skt_Write_Msg	×	Message Configuration
Configuration* Communication Tag		Configuration* Communication* Tag
Message Type: CIP Generic Service WriteSocket Service Generic Service (Hex) Class: 342 (Hex) Code: 4e (Hex) Class: 342 (Hex) Instance: 0 Attribute: 0 (Hex)	Source Element: EKS_Milling.WriteSrc Source Length: 1 1 - Bestination EKS_Milling.WriteSize Element: EKS_Milling.WriteSizeSen	Path: THIS THIS THIS Broadcast: Communication Method
	- Netra Ogina	CIP With Source ID Source Link: D Destination Node: D (Octal) Connected Cache Connections Curve Connection
C Enable C Enable Waiting C Start	O Done Done Length: 0	○ Enable ○ Enable ○ Start ○ Done Done Length: 0 ○ Error Code: Extended Error Code: □ Timed Out ◆
Error Text:	Abbrechen Übernehmen Hilfe	Error Path: Error Text: OK Abbrechen Ubernehmen Hiffe

Fig. 16: Configuring parameters for *WriteSocket* Message

6. Open the Configuration Dialog for SktDelMsg. Select Service Type: DeleteSocket. Configure the following parameters:

Parameter	value	
Instance	0	
Path	THIS (you will find the Path parameter on the Commun	ication tab)
Message Configuration - EKS_Skt	_Del_Msg	Message Configuration
Configuration* Communication	Tag	Configuration* Communication* Tag
Message Type: CIP Gene	eric 🔻	<u>P</u> ath: THIS <u>B</u> rowse
Service DeleteSocket Type: Service 4f (Hex) Class: Code: 0 Attribute:	Source Element: ▼ Source Length: 0 ▲ (Bytes) 342 (Hex) Destination Element: ▼ 0 (Hex)	THIS Broadcast: Communication Method © CIP DH+ DH+ Destination Link: CIP With Source Link: Source ID Source Link: Connected Cache Connections
Enable C Enable Waiting Error Code: Extense Error Path: Error Text:	O Start O Done Done Length: 0 ded Error Code: □ Timed Out ←	O Enable ◯ Enable Waiting ◯ Start ◯ Done Done Length: 0 O Error Code: Extended Error Code: □ Timed Out ◆ Error Path: Error Text: OK Abbrechen Übernehmen Hilfe

Fig. 17: Configuring parameters for DeleteSocket Message

5.4. Activating socket connection

Create a new rung. Add an input variable (e.g. *EnableSocket*) to the rung as a contact (*Examine On*). Add the variable*Skt*-*ConEnable* from the AOI as an output (*Output Energize*).

💰 Logia	c Desig	ner - AP000250 [1	1756-L81ES 32.12]			- 🗆 ×				
📙 Mair	Progr	ram - MainRoutin	ie* ×			-				
•			abrd ab (ab)							
		EnableSocket		[AOI_EKS_Mil	illing.SktConEnable 🗸 🔨				
0 2	9					Enter Name Filter		~	Show: All Tags	~
1			AOI EKS TCPIP V32 Y AOI EKS TCPIP V32 Y SKConEnable SKEKSData SktTimeout SktCreateMsg SktConnectMsg SktWinteMsg SktWiteMsg SktWiteMsg	YYYMMDD YY AOI_EKS_Milling EKS_Milling 2000 EKS_Skt_Create_Msg EKS_Skt_Connect_Msg EKS_Skt_Write_Msg EKS_Skt_Write_Msg EKS_Skt_Write_Msg EKS_Skt_Del_Msg	-(SktConT -(SktUsed -(SktError -(SktCpen -(SktRead	Name AOLEKS_Milling Show controller tags	inableIn inableOut iktConToServ iktConEnable iktUsed iktError Data Desc	AOI_EKS BOOL BOOL BOOL BOOL BOOL BOOL HE: AOI_EKS_I Type: BOOL tription: Enal	Milling.SktConEnable	^ ^
			JobFinishedActiveTime EKSStartAddressRead EKSNumberOfBytesRead	500 ← 0	–(SktWrite –(EKSKey –(EKSStat	^{le} ✓ Show MainProgram tags ^y Show parameters from <u>o</u> the at <none></none>	r program:	~		I Show s <u>a</u> fety tags
			EKSKeyDataRead EKSStartAddressWrite EKSNumberOfBytesWrite	EKS_Key_Data_Read 0 116	—(JobFinis	shea ,				
			EKSKeyDataWrite EKSStatusNumber	EKS_Key_Data_Write 16#0000						

Fig. 18: Rung for activating the socket connection

5.5. Command for writing Electronic-Key

Create another rung and add an input variable (e.g. WriteKey) as a contact. Add the variable *EKSWriteKeyCommand* from the AOI as an output.

Fig. 19: Rung for writing Electronic-Key

5.6. Configuring the parameters for the IP address

TIP

The IP address of the Electronic-Key adapter EKS is assigned using the web interface. You will find the related description in the manual, chapter 7.2.

Open the Controller Tags and change the view to Monitor Tags. Select the instance EKS_Milling and enter the IP address as a string.

Controller Tags - AP000250(controll	er) ×	
cope: PAP000250 V Show	r. All Tags	
Name	≡≡ - Value	
Local:3:0		(
Local:3:I	String Browser - EKS Milli	ing.IPAdress* X
Local:3:C		
Local:2:1	192.168.0.220	^
Local:2:C		a
EKS_Skt_Write_Msg		\$
EKS_Skt_Read_Msg		\$
EKS_Skt_Del_Msg		4
EKS_Skt_Create_Msg		~
EKS_Skt_Connect_Msg	OK Cancel	I Apply Help
 EKS_Milling 	and 0 Error(s)	13 INS 13 of 82
EKS_Milling.IPAdress	[····	
EKS_Milling.Inst		
EKS_Milling.WriteSizeSent		
EKS_Milling.CreateSrc		
EKS_Milling.OpenSrc		
EKS_Milling.ReadSrc		
EKS_Milling.ReadResponse		
N EVS Milling WriteSrc		

Fig. 20: Entering the IP address

5.7. Starting the program

Load the program into the control system. Go *Online* and set the bit *EnableSocket* (=*TRUE*). The socket connection is open if the output bit *SktOpen* for the AOI is *TRUE*. If an Electronic-Key is placed in the Electronic-Key adapter, the output bit *EKSKeyIN* = *TRUE* and the data read are saved in the *EKS_Key_Data_Read* array.

Fig. 21: Socket connection established

🖉 Controller Tags - AP000250(controller) 🗙							
Scope:	PAP000250 ~	Show:	All Tags				
Name	•	-8 -	Value 🗧	Force Mask 🔹 🕈	Style	Data Type	
▲ EKS	_Key_Data_Read		{}	{}	ASCII 🗸	SINT[124]	
♦ El	<pre>Key_Data_Read[0]</pre>		'E'		ASCII	SINT	
♦ El	<pre>Key_Data_Read[1]</pre>		'U'		ASCII	SINT	
♦ El	<pre>Key_Data_Read[2]</pre>		'C'		ASCII	SINT	
► EI	KS_Key_Data_Read[3]		.н.		ASCII	SINT	
♦ El	<pre>Key_Data_Read[4]</pre>		'N'		ASCII	SINT	
► EI	<pre>Key_Data_Read[5]</pre>		'E'		ASCII	SINT	
► El	<pre>Key_Data_Read[6]</pre>		'R'		ASCII	SINT	
► EI	<pre>Key_Data_Read[7]</pre>				ASCII	SINT	
► EI	KS_Key_Data_Read[8]		T		ASCII	SINT	
► EI	<pre>Key_Data_Read[9]</pre>		'E'		ASCII	SINT	
♦ El	<pre>Key_Data_Read[10]</pre>		.A.		ASCII	SINT	
▶ El	<pre>Key_Data_Read[11]</pre>		'E'		ASCII	SINT	
▶ El	<pre>Key_Data_Read[12]</pre>		τ		ASCII	SINT	
♦ El	KS_Key_Data_Read[13]				ASCII	SINT	
▶ El	<pre>Key_Data_Read[14]</pre>		'3'		ASCII	SINT	
▶ El	<s_key_data_read[15]< td=""><th></th><td>'\$00'</td><td></td><td>ASCII</td><td>SINT</td></s_key_data_read[15]<>		'\$00'		ASCII	SINT	
▶ El	KS_Key_Data_Read[16]		'\$00'		ASCII	SINT	
♦ El	KS_Key_Data_Read[17]		'\$00'		ASCII	SINT	
♦ El	KS_Key_Data_Read[18]		'\$00'		ASCII	SINT	
♦ El	KS_Key_Data_Read[19]		'\$00'		ASCII	SINT	
▶ El	<pre>Key_Data_Read[20]</pre>		' \$00'		ASCII	SINT	
I → \ M	onitor Tags (Edit Tags	s /			<		

Fig. 22: Result in EKS_Key_Data_Read array

5.8. Writing the Electronic-Key

To write an Electronic-Key, the *EKS_Key_Data_Write* array must be filled with the data to be written. As the Electronic-Key is always written in 4-byte blocks, all bytes into which you have not entered any values will be overwritten with 0. If you want to change individual data items on the Electronic-Key, you must always write the data that are not to be changed to the array as well. In this example, we will change stage 3 to stage 5.

Controller Tags - AP000250(controller) ×					
Scope: AP000250 V Sho	w: All Tags				
Name	∎ - Value	+ Forc	e Mask 🔹 🕈	Style	Data Type
EKS_Key_Data_Write		{}	{}	ASCII	SINT[116]
EKS_Key_Data_Write[0]		'E'		ASCII	SINT
EKS_Key_Data_Write[1]		'U'		ASCII	SINT
EKS_Key_Data_Write[2]		'C'		ASCII	SINT
EKS_Key_Data_Write[3]		Ή.		ASCII	SINT
EKS_Key_Data_Write[4]		'N'		ASCII	SINT
EKS_Key_Data_Write[5]		'E'		ASCII	SINT
EKS_Key_Data_Write[6]		'R'		ASCII	SINT
EKS_Key_Data_Write[7]				ASCII	SINT
EKS_Key_Data_Write[8]		τ		ASCII	SINT
EKS_Key_Data_Write[9]		'E'		ASCII	SINT
EKS_Key_Data_Write[10]		·v·		ASCII	SINT
EKS_Key_Data_Write[11]		'E'		ASCII	SINT
EKS_Key_Data_Write[12]		T		ASCII	SINT
EKS_Key_Data_Write[13]				ASCII	SINT
EKS_Key_Data_Write[14]	\sim	'5'		ASCII	SINT
EKS_Key_Data_Write[15]		'\$00'		ASCII	SINT
EKS_Key_Data_Write[16]		'\$00'		ASCII	SINT
EKS_Key_Data_Write[17]		'\$00'		ASCII	SINT
EKS_Key_Data_Write[18]		'\$00'		ASCII	SINT
EKS_Key_Data_Write[19]		'\$00'		ASCII	SINT
EKS_Key_Data_Write[20]		'\$00'		ASCII	SINT
♦ Monitor Tags / Edit Tags /					¢

Fig. 23: Filling the array EKS_Key_Data_Write

If you now want to write the data from the array to the Electronic-Key, the *WriteKey* bit will require an edge. The data are written to the Electronic-Key and are then immediately displayed in the *EKS_Key_Data_Read* array.

Fig. 24: Writing data to the Electronic-Key

Controller Tags - AP000250(controller) ×						
Scope:	PAP000250 ~	Show:	All Tags			
Name	,	== -	Value	Force Mask 🔹	Style	Data Type
▲ EKS	_Key_Data_Read		{	} {}	ASCII	SINT[124]
► EF	(S_Key_Data_Read[0]		'E	r	ASCII	SINT
► EF	KS_Key_Data_Read[1]		ι. U	r.	ASCII	SINT
► EF	(S_Key_Data_Read[2]		'C		ASCII	SINT
► EF	KS_Key_Data_Read[3]		۲	r	ASCII	SINT
► EF	(S_Key_Data_Read[4]		'N	r	ASCII	SINT
► EF	KS_Key_Data_Read[5]		'E		ASCII	SINT
► EF	(S_Key_Data_Read[6]		'F	r -	ASCII	SINT
► E	KS_Key_Data_Read[7]			•	ASCII	SINT
► EF	(S_Key_Data_Read[8]		1	:	ASCII	SINT
► EF	(S_Key_Data_Read[9]		'E	2	ASCII	SINT
► EF	KS_Key_Data_Read[10]		1	r	ASCII	SINT
► EF	(S_Key_Data_Read[11]		'E		ASCII	SINT
► EF	KS_Key_Data_Read[12]		1	:	ASCII	SINT
► EF	(S_Key_Data_Read[13]				ASCII	SINT
▶ EF	KS_Key_Data_Read[14]		Ĭ [™] []	ASCII	SINT
► EF	(S_Key_Data_Read[15]		\$00	r	ASCII	SINT
► EF	(S_Key_Data_Read[16]		'\$00	r	ASCII	SINT
► EF	KS_Key_Data_Read[17]		'\$00	r	ASCII	SINT
► EF	(S_Key_Data_Read[18]		\$00	r	ASCII	SINT
► EF	KS_Key_Data_Read[19]		\$00	r	ASCII	SINT
► EF	(S_Key_Data_Read[20]		\$00	r	ASCII	SINT
Monitor Tags / Edit Tags /						

Fig. 25: New Electronic-Key data

6. Important note – 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 technology 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.

EN

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

Edition: AP000250-03-04/21 Title: Application EKS Integration of EKS with TCP/IP Interface in Rockwell Studio 5000®

Copyright: © EUCHNER GmbH + Co. KG, 04/2021

Subject to technical modifications; no responsibility is accepted for the accuracy of this information. $% \label{eq:sub_constraint}$