

# Electronic-Key-System

## Manual

### Ethernet ActiveX<sup>®</sup> Module Software

Order No. 102 030



# EKS.



More than safety.



# EUCHNER

## Table of contents

<b>1</b>	<b>General notes.....</b>	<b>4</b>
1.1	Use of the manual.....	4
1.2	Explanation of symbols.....	4
1.3	Requirements on the user .....	4
1.4	System requirements.....	4
<b>2</b>	<b>Support information, installing and uninstalling .....</b>	<b>5</b>
<b>3</b>	<b>The EKS Ethernet ActiveX® module .....</b>	<b>6</b>
3.1	EKS type library .....	6
3.2	EKS control.....	6
3.3	Overview of the methods, properties and events in the EKS Ethernet ActiveX® module .....	7
3.4	Methods .....	8
3.4.1	Open .....	8
3.4.2	Close .....	8
3.4.3	Read.....	8
3.4.4	Write.....	8
3.4.5	getData.....	9
3.4.6	setData.....	9
3.5	Properties.....	9
3.5.1	Port.....	9
3.5.2	IPAddress.....	10
3.5.3	KeyType .....	10
3.5.4	LastState (ReadOnly) .....	11
3.5.5	StartAdress .....	12
3.5.6	CountData .....	12
3.5.7	Opening.....	13
3.5.8	Reading.....	13
3.5.9	Writing .....	13
3.5.10	KeyState.....	13
3.5.11	Version .....	13
3.5.12	Data.....	14
3.5.13	Debug.....	14
3.6	Constants.....	15
3.7	Events .....	16
3.7.1	OnKey .....	16

---

3.7.2	OnRead.....	16
3.7.3	OnWrite.....	16
<b>4</b>	<b>Examples.....</b>	<b>17</b>
4.1	Establishing connection with EKS Electronic-Key adapter.....	17
4.2	Example event call in Visual Basic® .....	18

## 1 General notes

This ActiveX<sup>®</sup> module supports the integration of the Electronic-Key-System (EKS) Electronic-Key adapter with Ethernet interface into your PC application. EKS can thus be used, e. g., in conjunction with process visualization software. Data communication takes place using the TCP/IP protocol. The ActiveX<sup>®</sup> module is used here as a protocol driver.

With the aid of the EKS Ethernet ActiveX<sup>®</sup> module, communication can be straightforwardly established with the EUCHNER Electronic-Key-System (EKS) from programming environments that support ActiveX (e. g. Microsoft Visual Basic<sup>®</sup>) or user programs (e. g. Microsoft Excel<sup>®</sup>). For this purpose the ActiveX<sup>®</sup> module must be installed and integrated into the related programming environment.

### 1.1 Use of the manual

This manual explains the functions of the EKS Ethernet ActiveX<sup>®</sup> module (order no. 100 665), from version 1.0.0.0.

### 1.2 Explanation of symbols

The following symbols are used in this manual to identify important instructions and useful information:



**Information!**

Important information is provided to the user here.



**Warning!**

Risk of loss of data.

### 1.3 Requirements on the user

To be able to use the EKS Ethernet ActiveX<sup>®</sup> module correctly, you must have knowledge of the utilization of ActiveX<sup>®</sup> modules. To be able to straightforwardly integrate the EKS hardware into your overall system, you must have read and understood the manual for the Electronic-Key adapter.

### 1.4 System requirements

**Hardware:** Standard PC, with network connection

**Software:** TCP-IP protocol must be installed.

**Operating system:** Windows<sup>®</sup> 2000  
Windows<sup>®</sup> XP

## 2 Support information, installing and uninstalling

To be able to use the EUCHNER EKS Ethernet ActiveX® module, you must first install it. For this purpose run the program EKS\_Ethernet\_ActiveX\_Module.exe.

**Information!**

During the installation you will be prompted to enter an installation folder. Once the installation is complete this folder will contain:

- ▶ the ActiveX® module ekseth.ocx
- ▶ this manual in Acrobat PDF format

The installation CD contains:

- ▶ the program EKS\_Ethernet\_ActiveX\_Module.exe
- ▶ this manual in Acrobat PDF format
- ▶ programming examples for various programming environments

To uninstall the ActiveX® module or to obtain support information, proceed as follows:

1. In the operating system select *Settings | Control Panel | Add/Remove Programs*.
2. In the list of programs installed select the entry *EUCHNER EKS Ethernet ActiveX Module*. You can also display support information here.

**Information!**

Always have the support information at hand when contacting EUCHNER.

3. To uninstall, click *Change/Remove* and follow the instructions in the uninstall dialog box.

## 3 The EKS Ethernet ActiveX<sup>®</sup> module

### 3.1 EKS type library

- ▶ Description EUCHNER EKS Ethernet ActiveX module
- ▶ Library EKSEthLib
- ▶ File name ekseth.ocx
- ▶ GUID { 36B62232-F5C4-4b46-BA4A-4B1F3F2C7974 }
- ▶ Control EKSETH

### 3.2 EKS control

- ▶ Control name EKSETH
- ▶ File name ekseth.ocx
- ▶ GUID { 72484DED-F77A-487c-9DC7-5751D10DBF17 }
- ▶ Methods 6
- ▶ Properties 13
- ▶ Events 3

Before you can use the EKS Ethernet ActiveX<sup>®</sup> module in your application, you must add the file ekseth.ocx to your project. To use an application that makes use of the ActiveX<sup>®</sup> module, you must install the module on your computer.

### 3.3 Overview of the methods, properties and events in the EKS Ethernet ActiveX® module

The EKS Ethernet ActiveX® module contains methods, properties and events that can be integrated into your programming environment.

- ▶ **Methods** are used for establishing the connection and transferring data between the user program and the EKS Electronic-Key adapter.
- ▶ **Properties** are used for settings, reflect states and contain data read from the Electronic-Key or that are to be written to the Electronic-Key.
- ▶ **Events** report the completion of a method or signal an event (e. g. Electronic-Key inserted).

All methods, properties and events for the EKS object are listed in the following table.

<b>Methods</b>	<b>Page</b>
Open	8
Close	8
Read	8
Write	8
getData	9
setData	9
<b>Properties</b>	<b>Page</b>
Port	9
IPAddress	10
KeyType	10
LastState	10
StartAddress	12
CountData	12
Opening	13
Reading	13
Writing	13
KeyState	13
Version	13
Data	14
Debug	14
<b>Events</b>	<b>Page</b>
OnKey	16
OnRead	16
OnWrite	16

## 3.4 Methods

### 3.4.1 Open

- ▶ Description Opens the connection to the EKS with the properties set (*IPAddress*, *Port*, *KeyType*, *StartAddress*, *CountData* ...).
- ▶ Syntax *Boolean* = *object*.**EKS.Open**;
- ▶ Comments The EKS must be connected and ready for operation before this method is used. The method returns the value *True* (error-free execution) or *False* (status message has been generated). If a status message has been generated the cause can be determined using the property *LastState*. You will find an overview on the status messages for the ActiveX® module in 3.5.4. On completion of the asynchronous execution the event *OnKey* is triggered. To obtain the actual state of the method *Open*, the property *Opening* can be polled. At the end of a program any open connection must be closed again by calling the method *Close*.

### 3.4.2 Close

- ▶ Description Closes the connection to the EKS.
- ▶ Syntax *Boolean* = *object*.**Close ()**;
- ▶ Comments This method must be run at the end of the user program to clear the PC's interface.

### 3.4.3 Read

- ▶ Description Method for reading data from the Electronic-Key (start address is defined in the property *StartAddress* and the number of bytes of data in the property *CountData*)
- ▶ Syntax *Boolean* = *object*.**Read ()**;
- ▶ Comments If the method returns *True*, the data will be read from the EKS. These data are to be found in the property *Data* after the event *OnRead* is triggered. If *False* is returned it was not possible to start the read request without errors. In this case the status number is given in the property *LastState*. You will find an overview on the status messages for the ActiveX® module in 3.5.4.



#### Information!

If you only want to read the data on the Electronic-Key, you do not need to explicitly call the method *Read*. As soon as the event *OnKey* is triggered and the property *KeyState* = *EKS\_KEY\_IN*, the data on the actual key are available in the property *Data*. Prior to triggering the event *OnKey* the method *Read* is executed internally in the ActiveX® module.

### 3.4.4 Write

- ▶ Description Method for writing data to the Electronic-Key (start address is defined in the property *StartAddress* and the number of bytes of data in the property *CountData*)
- ▶ Syntax *Boolean* = *object*.**Write ()**;
- ▶ Comments If the method returns *True*, the data will be written to the Electronic-Key. The write request is complete after the event *OnWrite* is triggered. If *False* is returned it was not possible to start the write request without errors. In this case the status number is given in the property *LastState*. You will find an overview on the status messages for the ActiveX® module in 3.5.4

### 3.4.5 getData

- ▶ Description: Reading access to the internal memory area of the ActiveX<sup>®</sup> module in which data read by the method *Read* or the event *OnKey* are stored.
- ▶ Syntax: **short** = *object.getData* (*short* ByteIndex);
- ▶ Comments: With the method *getData*, the internal memory area of the ActiveX<sup>®</sup> module can be read. Upon triggering the events *OnRead* or *OnKey*, the key data are available in the internal memory and can be read using *getData*. In the properties *StartAddress* and *CountData*, the range is defined from which byte data shall be read (method *Read*).



#### Information!

This is an additional way of accessing the internal memory of the ActiveX<sup>®</sup> module. This method can be used in programming environments which do not offer array support. Usually, the internal memory area is accessed via the property *Data*, see 3.5.12.

### 3.4.6 setData

- ▶ Description: Writing access to the internal memory area of the ActiveX<sup>®</sup> module in which data to be written by the method *Write* are stored.
- ▶ Syntax: *object.setData* (*short* ByteIndex, *short* DataValue);
- ▶ Comments: With the method *setData*, the internal memory area of the ActiveX<sup>®</sup> module can be written. Upon triggering the event *OnWrite*, the data are written from the temporary storage to the key. In the properties *StartAddress* and *CountData*, the range is defined from which byte data shall be written (method *Write*).



#### Information!

This is an additional way of accessing the internal memory of the ActiveX<sup>®</sup> module. This method can be used in programming environments which do not offer array support. Usually, the internal memory area is accessed via the property *Data*, see 3.5.12.

## 3.5 Properties

### 3.5.1 Port

- ▶ Description: Selects the port for the TCP connection
- ▶ Syntax: *object.Port* = **String** Value;
- ▶ Comments: Possible values are:  
2444  
2445  
...  
This property is applied by calling the method *Open*. The value must match the setting on the EKS.
- ▶ Data type: **String**
- ▶ Default value: 2444

### 3.5.2 IPAddress

- ▶ Description Selects the IP address for the TCP connection
- ▶ Syntax *object.IPAddress = String* Value;
- ▶ Comments Possible values are:  
192.168.1.1  
...  
This property is applied by calling the method *Open*. The value must match the setting on the EKS.
- ▶ Data type **String**
- ▶ Default value 192.168.1.1

### 3.5.3 KeyType

- ▶ Description Defines the type of Electronic-Key used (read-only Electronic-Key\* or read/write Electronic-Key)  
\*Earlier transponder type. We do not recommend using this transponder type in new installations.
- ▶ Syntax *object.KeyType = KeyTypeConstants* Value;
- ▶ Comments Possible values are:  
EKS\_KEY\_READWRITE = 1  
EKS\_KEY\_READONLY = 8  
This property is applied by calling the method *Open*.
- ▶ Data type **KeyTypeConstants** (Enumeration)
- ▶ Default value EKS\_KEY\_READWRITE

### 3.5.4 LastState (ReadOnly)

- ▶ Description            Status of the last method called (0=OK or status number)
- ▶ Syntax                **long** = *object.LastState*;
- ▶ Comments            After a method is run (*Read*, *Write*, ...) or after an event (*OnKey*, *OnRead*, ...), you can determine here whether the method was run correctly. Status numbers in the range from 0 to 127 (0<sub>hex</sub> to 7F<sub>hex</sub>) are generated by the EKS and are documented in the manual for the EKS Electronic-Key adapter. Status numbers between 128 and 255 (80<sub>hex</sub> to FF<sub>hex</sub>) are generated by the ActiveX® module.
- ▶ Data type            **long**
- ▶ List of status numbers for the ActiveX® module:



#### Warning!

Immediately after a method has been called or an event has been triggered you should poll the value in the property *LastState*. Otherwise the property *LastState* could be overwritten by another method, as only the status message from the last method run is given in the property *LastState*. This warning also applies to internal methods that run in the background and that are not started by you.

Value		Description
hex	dec	
0x90	144	<b>WrongParam</b> The TCP port given in the property <i>Port</i> is not in the range >1024 und <65535
0xA0	160	<b>DeviceNotOpened</b> The connection to the EKS has not been opened, please run the method <i>Open</i> .
0xB0	176	<b>ReadTimeOut</b> It was not possible to correctly complete the method <i>Read</i> , the method has timed out.
0xB1	177	<b>WriteTimeOut</b> It was not possible to correctly complete the method <i>Write</i> , the method has timed out.
0xB2	178	<b>TimeOut</b> An internal method in the ActiveX® module has timed out.
0xC0	192	<b>NothingToRead</b> The number of bytes of data to be read, as defined by the property <i>CountData</i> , is 0.
0xC1	193	<b>NothingToWrite</b> The number of bytes of data to be written, as defined by the property <i>CountData</i> , is 0.
0xE0	224	<b>OpenFailed</b> The method <i>Open</i> failed.
0xE1	225	<b>OpenActive</b> The method <i>Open</i> is still active.
0xE8	232	<b>Suspend</b> The computer will be placed in the suspend mode.
0xE9	233	<b>ResumeSuspend</b> The suspend mode has been terminated.
0xEA	234	<b>ConnectionTimeOut</b> Connection timeout to the EKS. It was not possible to establish a connection to the EKS in the time specified.
0xEB	235	<b>ConnectionLost</b> The connection to the EKS has been interrupted.
0xEC	236	<b>Reconnect</b> The connection to the EKS has been re-established.
0xFF	255	<b>Busy</b> The ActiveX® module is busy processing a method, the request cannot be run.

### 3.5.5 StartAdress

- ▶ Description The start address for the memory area on the Electronic-Key from which data are to be read (*Read*) respectively from which data are to be written (*Write*).
- ▶ Syntax *object.StartAdress = short* Value;
- ▶ Comments Defines the start address for the data to be read using the method *Read* as well as the start address for the data to be written using the method *Write*. Once the read method has been completed successfully, the data will be available in the property *Data*. The data to be written must also be saved there. The property *StartAdress* must be set prior to calling the method so that the start address can be used for the subsequent call.



#### Information

On Electronic-Keys read/write with 116 bytes, the memory is organized in 4-byte blocks. This means a multiple of 4-byte sized blocks must always be written.

The start address must be given in the range byte number 0 to byte number 112, always in 4-byte steps (byte number 0, 4, 8 ... 112)!

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 contains a unique 8-byte serial number. This number is written by laser during the Electronic-Key production process and can never be changed or deleted. The serial number is used for secure distinction of every single Electronic-Key. It is necessary to completely evaluate all 8 bytes. The serial number is appended to the freely programmable user data (from byte number 116).

- ▶ Data type **short**
- ▶ Default value 0

### 3.5.6 CountData

- ▶ Description The number of bytes of data to be written or read.
- ▶ Syntax *object.CountData = short* Value;
- ▶ Comments Defines the number of bytes of data to be read using the method *Read* as well as the number of bytes of data to be written using the method *Write*. Once the read method has been completed successfully, the data will be available in the property *Data*. The data to be written must also be saved there. The property *CountData* must be set prior to calling the methods so that the number of bytes of data to be read/written can be used for the subsequent call.



#### Information!

On Electronic-Keys read/write with 116 bytes, the memory is organized in 4-byte blocks. This means a multiple of 4-byte sized blocks must always be written.

The start address must be given in the range byte number 0 to byte number 112, always in 4-byte steps (byte number 0, 4, 8 ... 112)!

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 contains a unique 8-byte serial number. This number is written by laser during the Electronic-Key production process and can never be changed or deleted. The serial number is used for secure distinction of every single Electronic-Key. It is necessary to completely evaluate all 8 bytes. The serial number is appended to the freely programmable user data (from byte number 116).

- ▶ Data type **short**
- ▶ Default value 4

### 3.5.7 Opening

- ▶ Description State of the method *Open*
- ▶ Syntax **bool = object.Opening;**
- ▶ Comments If the property *Opening* returns the value *True*, the method *Open* is currently active. As long as this method is active it is not possible to call any other methods
- ▶ Data type **bool**
- ▶ Default value false

### 3.5.8 Reading

- ▶ Description State of the method *Read*
- ▶ Syntax **bool = object.Reading;**
- ▶ Comments If the property *Reading* returns the value *True*, the method *Read* is currently active. The data on the Electronic-Key are not yet available in the property *Data*. As long as this method is active it is not possible to call any other methods.
- ▶ Data type **bool**
- ▶ Default value false

### 3.5.9 Writing

- ▶ Description State of the method *Write*
- ▶ Syntax **bool = object.Writing;**
- ▶ Comments If the property *Writing* returns the value *True*, the method *Write* is currently active. The write request is still active and the data have not yet been completely written to the Electronic-Key. As long as this method is active it is not possible to call any other methods.
- ▶ Data type **bool**
- ▶ Default value false

### 3.5.10 KeyState

- ▶ Description Returns the status of the last event.
- ▶ Syntax **bool = object.KeyState;**
- ▶ Comments Possible parameters are:
  - ▶ EKS\_KEY\_IN = 1
  - ▶ EKS\_KEY\_OUT = 2
  - ▶ EKS\_KEY\_OTHER = 3
- ▶ Data type **KeyStateConstants** (Enumeration)
- ▶ Default value EKS\_KEY\_OUT

### 3.5.11 Version

- ▶ Description Returns the actual version of the EKS Ethernet ActiveX® module
- ▶ Syntax **String Value = object.Version;**
- ▶ Data type **String**

### 3.5.12 Data

- ▶ Description Memory area in which data read by the method *Read* or the event *OnKey*, or data to be written using the method *Write*, are stored.
- ▶ Syntax **short = object.Data (short ByteIndex);**
- ▶ Comments The property *Data* represents a cache for all data that are read from the Electronic-Key and that are to be written to the Electronic-Key. The data for the Electronic-Key are provided or assigned in bytes. After the event *OnRead* or *OnKey* is triggered, the data on the Electronic-Key are available in the property *Data*. Once the event *OnWrite* has been triggered the data have been written from the property *Data* to the Electronic-Key. In the properties *StartAdress* and *CountData*, the range is defined from which byte data shall be read (method *Read*) respectively data shall be written (method *Write*).
- ▶ Data type **short**
- ▶ Default value -12851 (CDCD<sub>hex</sub>)  
The default value is present if no data have been read from the Electronic-Key or there is no Electronic-Key in the Electronic-Key adapter.

### 3.5.13 Debug

- ▶ Description If the property *Debug* is set to the value *true*, the method *Close* is called for all instances at the end of a debug session in a programming environment.
- ▶ Syntax **bool = object.Debug;**
- ▶ Comments The ActiveX<sup>®</sup> module is not correctly destructed at the end of the debug session in some programming environments. The property *Debug* must, e. g., be set to *true* in Microsoft Visual Basic<sup>®</sup> and Microsoft Excel<sup>®</sup> for the development of the application so that the TCP connection is closed at the end of the debug session without explicitly calling the method *Close*. If a debug session is terminated before the method *Close* is called, the connection to the EKS remains open. If the property *Debug* is *true*, then the TCP connections for **all** ActiveX<sup>®</sup> instances are closed at the end of the debug session.
- ▶ Data type **bool**
- ▶ Default value **false**

#### Information!



Please ensure you only set the property *Debug* to *true* during the debug session. If **one** control is destructed on the use of several instances of the ActiveX<sup>®</sup> module, the connection to the EKS will be closed in all other instances.

### 3.6 Constants

This section lists all constants that are used in the properties for the EKS Ethernet ActiveX® module. The constants are also listed in the description of the properties and methods in which they are used.

#### ***KeyStateConstants*** (used in the property *KeyState*)

Value	Constant
1	EKS_KEY_IN
2	EKS_KEY_OUT
3	EKS_KEY_OTHER

#### ***KeyTypeConstants*** (used in the property *KeyType*)

Value	Constant
1	EKS_KEY_READWRITE
8	EKS_KEY_READONLY

## 3.7 Events

### 3.7.1 OnKey

- ▶ Description This event must be defined in the user program and is called by the ActiveX® module.
- ▶ Syntax **Private Sub** *object\_OnKey* ()
- ▶ Comments To use this event, a method with the name *OnKey* must be defined in the user program. This method is called by the ActiveX® module as soon as there is a change in the EKS (Electronic-Key inserted /Electronic-Key removed, etc.). The user can then poll which event has occurred in the user program (EKS\_KEY\_IN, EKS\_KEY\_OUT, EKS\_KEY\_OTHER). The event *OnKey* with the property *KeyState=EKS\_EVENT\_KEYIN* is triggered when there is a new Electronic-Key in the EKS. The user can then read the data on the key from the property *Data* **without calling the method Read**. The event *OnKey* with the property *KeyState=EKS\_KEY\_OUT* is triggered on the removal of the Electronic-Key. The event *OnKey* with the property *KeyState=EKS\_KEY\_OTHER* is triggered when the ActiveX® module detects a status. The related status number can be read in the property *LastState*.

#### Information!



Whether an Electronic-Key has been inserted or removed is detected from a network telegram (see manual for EKS Electronic-Key adapter). The telegram is sent immediately after insertion or removal of the Electronic-Key from the key adapter to the presently connected ActiveX® client.

### 3.7.2 OnRead

- ▶ Description This event must be defined in the user program and is called by the ActiveX® module.
- ▶ Syntax **Private Sub** *object\_OnRead* ()
- ▶ Comments To use this event a method with the name *OnRead* must be defined in the user program. This method is called by the ActiveX® module as soon as the method *Read* is completed in the ActiveX® module. The related status number can be read in the property *LastState*.

### 3.7.3 OnWrite

- ▶ Description This event must be defined in the user program and is called by the ActiveX® module.
- ▶ Syntax **Private Sub** *object\_OnWrite* ()
- ▶ Comments To use this event a method with the name *OnWrite* must be defined in the user program. This method is called by the ActiveX® module as soon as the method *Write* is completed in the ActiveX® module. The related status number can be read in the property *LastState*.

## 4 Examples

**Information!**

The installation CD contains examples for the integration of the EKS Ethernet ActiveX<sup>®</sup> module in various programming environments.

### 4.1 Establishing connection with EKS Electronic-Key adapter

The following example shows how the method *Open* can be used. The values shown correspond to the default settings for the properties. It may be necessary to change these values for your application.

1. Set required values in the properties. These settings can also be made in the programming tool (e.g. Visual Basic<sup>®</sup>) using the properties of the object *EKS*:

```
EKS.Port = "2444"  
EKS.KeyType = EKS_KEY_READWRITE
```

2. Set the required read/write parameters (can also be set after opening the interface):

```
EKS.StartAdress = 0  
EKS.CountData = 4
```

3. Open interface:

```
EKS.Open
```

**Information!**

If the default values shown are used, it is sufficient to use just the one line with the call *EKS.Open*.

## 4.2 Example event call in Visual Basic®

```
Private Sub EKS_OnKey( )
    Select Case KeyState
        Case EKS_EVENT_KEYIN
            User functions KeyIn
            ' e. g. Read key data from the EKS Electronic-Key
            ' Attention! It is not necessary to call the Read method!
            for i=0 to 116
                KeyData = KeyData & EKS.Data(i)
            Next i
        Case EKS_EVENT_KEYOUT
            User functions KeyOut
            ' e. g. Delete Electronic-Key data in the user software
            KeyData = "-"
        Case EKS_EVENT_OTHER
            User functions Other
            ' e. g. Poll status number
            StatusNumber = EKS.LastState
    End Select
End Sub
```



Microsoft Windows<sup>®</sup>, Excel<sup>®</sup>, ActiveX<sup>®</sup>  
and Visual Basic<sup>®</sup> are registered  
trademarks of Microsoft Corporation

More than safety.



EUCHNER GmbH + Co. KG  
Kohlhammerstraße 16  
D-70771 Leinfelden-Echterdingen

Telephone +49 711 / 75 97 - 0  
Fax +49 711 / 75 33 16  
[www.euchner.de](http://www.euchner.de) · [info@euchner.de](mailto:info@euchner.de)

**EUCHNER**