Electronic-Key-System

Manual

Software, ActiveX® Module Ethernet TCP/IP

Document no. 2102030

EUCHNER

More than safety.

Table of contents

1 General notes

This ActiveX® 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 in conjunction with process visualization software, for example. Data communication takes place using the TCP/IP protocol. The ActiveX® module is used here as a protocol driver.

With the aid of the EKS Ethernet ActiveX® module, communication can be straightforwardly established with the EUCHNER Electronic-Key System EKS from programming environments that support ActiveX (e.g. Microsoft Visual Basic®) or user programs (e.g. Microsoft Excel®). For this purpose, the ActiveX® 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 ActiveX® moduls Ethernet TCP/IP (Software order no. 8100665) Version 1.0.X.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.

Attention!
Risk of loss of data.

1.3 Requirements on the user

To be able to use the EKS Ethernet ActiveX® module correctly, you must have knowledge of the utilization of ActiveX® 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® Server 2008, 32-bit
Windows® Server 2008, 64-bit
Windows® 7, 32-bit
Windows® 7, 64-bit
Windows® Server 2008, R2, 64-bit
Windows® 10, 32-bit
Windows® 10, 64-bit

2 Support information, installing and uninstalling

To be able to use the EUCHNER EKS Ethernet TCP/IP ActiveX® module, you must first install it. Run the installation file corresponding to the respective user program:

- For 32-bit user program: EKS_Ethernet_ActiveX_Module.msi
- For 64-bit user program: EKS_Ethernet_ActiveX_Module_x64.msi

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

- the ActiveX® module

The downloaded zip archive includes:

- the installation files EKS_Ethernet_ActiveX_Module.msi and EKS_Ethernet_ActiveX_Module_x64.msi
- 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® 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® module in your application, you must add the file ekseth.ocx to your project. To use an application that makes use of the ActiveX® 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.

Methods	**Chapter**
Open	3.4.1 Open
Close	3.4.2 Close
Read	3.4.3 Read
Write	3.4.4 Write
getData	3.4.5 getData
setData	3.4.6 setData
Properties	**Chapter**
Port	3.5.1 Port
IPAddress	3.5.2 IPAddress
KeyType	3.5.3 KeyType
LastState	3.5.4 LastState (ReadOnly)
StartAdress	3.5.5 StartAdress
CountData	3.5.6 CountData
Opening	3.5.7 Opening
Reading	3.5.8 Reading
Writing	3.5.9 Writing
KeyState	3.5.10 KeyState
Version	3.5.11 Version
Data	3.5.12 Data
Debug	3.5.13 Debug
Events	**Chapter**
OnKey	3.7.1 OnKey
OnRead	3.7.2 OnRead
OnWrite	3.7.3 OnWrite

3.4 Methods

3.4.1 Open

- Description: Opens the connection to the EKS with the properties set (*IPAddress, Port*, *KeyType*, *StartAdress*, *CountData* ...).
- Syntax: **Boolean** = *object*.**EKS.Open**;
- Remarks: 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 of the status messages for the ActiveX® module in chapter 3.5.4. On completion of asynchronous execution, the event *OnKey* is activated. 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*.

Information!
The method *Open* starts a background process to set up communication with the device. If *True* is returned, this indicates only that the background process could be started. It is not checked whether there is a physical connection to the device.

3.4.2 Close

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

3.4.3 Read

- Description: Method for reading data from the Electronic-Key (the start address is defined in the property *StartAdress* and the number of bytes of data in the property *CountData*)
- Syntax: **Boolean** = *object*.**Read ()**;
- Remarks: If the method returns *True*, the data are read from the EKS. These data can be found in the property *Data* after the event *OnRead* is activated. 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 of the status messages for the ActiveX® module in chapter 3.5.4.

Information!
If all you want to do is 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 activated and the property *KeyState* = *EKS_KEY_IN*, the data on the current Electronic-Key are available in the property *Data*. Prior to activation of the event *OnKey*, the method *Read* is performed internally in the ActiveX® module.

3.4.4 Write

- Description: Method for writing data to the Electronic-Key (the start address is defined in the property *StartAdress* and the number of bytes of data in the property *CountData*)
- Syntax: **Boolean** = *object*.**Write ()**;
- Remarks: If the method returns *True*, the data are written to the Electronic-Key. The write request is complete after the event *OnWrite* is activated. 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 chapter 3.5.4

3.4.5 getData

- Description: Read access to the internal memory of the ActiveX® module in which data read by the method *Read* or the event *OnKey* are saved.
- Syntax: **short** = *object*.**getData** (*short* ByteIndex);
- Remarks: The internal memory of the ActiveX® module can be read using the method *getData*. After the event *OnRead* or *OnKey* has been activated, the data on the Electronic-Key are available in the internal memory and can be read using *getData*. The properties *StartAdress* and *CountData* define from which byte data are to be read (method *Read*).

Information!
This is an additional way of accessing the internal memory for the ActiveX® module. This method can be used in programming languages that do not support arrays. The internal memory is normally accessed using the property *Data,* see chapter 3.5.12.

3.4.6 setData

- Description: Write access to the internal memory of the ActiveX® module in which the data to be written by the method *Write* are saved.
- Syntax: *object*.**setData** (*short* ByteIndex, *short* DataValue);
- Remarks: The internal memory of the ActiveX® module can be written using the method *setData*. Once the event *OnWrite* has been activated, the data are written from the clipboard to the Electronic-Key. The properties *StartAdress* and *CountData* define from which byte data are to be written (method *Write*).

Information!
This is an additional way of accessing the internal memory for the ActiveX® module. This method can be used in programming languages that do not support arrays. The internal memory is normally accessed using the property *Data,* see chapter 3.5.12.

3.5 Properties

3.5.1 Port

- Description: Selects the port for the TCP connection
- Syntax: *object*.**Port = String** Value;
- Remarks: 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;
- Remarks: 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 Electronic-Key type used. Only the read-/write key is supported in the EKS.
- Syntax: *object*.**KeyType = KeyTypeConstants** Value;
- Remarks: The value is:
  EKS_KEY_READWRITE = 1
  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**;
- Remarks: 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_{hex} to $7F_{hex}$) are generated by the EKS and are documented in the manual for the EKS Electronic-Key adapter. Status numbers between 128 and 255 (80_{hex} to FF_{hex}) are generated by the ActiveX® module.
- Data type: **long**
- List of status numbers for the ActiveX® module:

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

3.5.5 StartAdress

- Description: The start address for the memory on the Electronic-Key from which data are to be read (*Read*) or to which data are to be written (*Write*).
- Syntax: *object*.**StartAdress = short** Value;
- Remarks: 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 methods so that the start address can be used for the subsequent call.

Information!
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 from 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 safe differentiation of every single Electronic-Key. It is necessary that all 8 bytes are completely evaluated for safe 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.

- 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;
- Remarks: Defines the number of bytes of data to be read using the method *Read* as well as the number of bytes 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 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 from 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 safe differentiation of every single Electronic-Key. It is necessary that all 8 bytes are completely evaluated for safe 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.

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

3.5.7 Opening

- Description: State of the method *Open*
- Syntax: **bool =** *object*.**Opening**;
- Remarks: 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**;
- Remarks: 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**;
- Remarks: 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**;
- Remarks: 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 current version of the EKS Ethernet ActiveX® module
- Syntax: **String** Value = *object*.**Version**;
- Data type: **string**

3.5.12 Data

- Description: Memory in which data read by the method *Read* or the event *OnKey*, or data to be written using the method *Write*, are saved.
- Syntax: **short =** *object*.**Data** (**short** ByteIndex);
- Remarks: 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 Electronic-Key data are provided or assigned byte by byte. The Electronic-Key data are available in the property *Data* after activation of the event *OnRead* or *OnKey*. Once the event *OnWrite* has been activated, the data have been written from the property *Data* to the Electronic-Key. The properties *StartAdress* and *CountData* define from which byte data are to be read (method *Read*) or written (method *Write*).
- Data type: **short**
- Default value: -12851 ($CDCD_{hex}$)
  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**;
- Remarks: The ActiveX® module is not correctly destroyed at the end of the debug session in some programming environments. The property *Debug* must be set to the value *true*, e.g. in Microsoft Visual Basic® and Microsoft Excel®, for development of the application so that the TCP port 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*, the TCP connections of **all** ActiveX® instances will be closed at the end of the debug session.
- Data type: **bool**
- Default value: false

Information!
Please ensure you set the property *Debug* to *true* only during the debug session. If **one** control is destroyed on the use of several instances of the ActiveX® 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 of 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 ()
- Remarks: 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 activated when there is a new Electronic-Key in the EKS. The user can then read the data on the Electronic-Key from the property *Data* **without calling the method *Read***. The event *OnKey* with the property *KeyState=EKS_KEY_OUT* is activated on removal of the Electronic-Key. The event *OnKey* with the property *KeyState=EKS_KEY_OTHER* is activated when the ActiveX® module detects a status. The related status number can be read in the property *LastState*.

Information!
A network telegram indicates whether an Electronic-Key was inserted or removed (see manual for Electronic-Key adapter EKS Ethernet TCP/IP). The Electronic-Key adapter sends the telegram to the currently connected ActiveX® client as soon as the Electronic-Key is inserted or removed.

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 ()**
- Remarks: 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 ()**
- Remarks: 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 zip archive contains examples for integrating the EKS ActiveX® module in various programming environments.

4.1 Establishing connection with the 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®) 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 data from the EKS Electronic-Key
            ' ATTENTION! It is not necessary to call the Read method!
            for i=0 to 123
                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
```

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

Phone +49 711 / 75 97 - 0
Fax +49 711 / 75 33 16
www.euchner.com . info@euchner.de

EUCHNER
More than safety.