Electronic-Key-System

Handbuch

Software ActiveX®-Modul Ethernet TCP/IP

Best. Nr. 102 030

More than safety.

EUCHNER

Inhaltsverzeichnis

1 Allgemeine Hinweise

Die Anbindung der Electronic-Key-System (EKS) Schlüsselaufnahme mit Ethernet-Schnittstelle in Ihre PC-Applikation wird durch dieses ActiveX®-Modul unterstützt. So kann EKS z. B. in Verbindung mit Software zur Prozessvisualisierung eingesetzt werden. Die Datenkommunikation läuft über das TCP/IP Protokoll. Das ActiveX®-Modul dient dabei als Protokolltreiber.

Mit Hilfe des EKS Ethernet ActiveX®-Moduls ist es einfach, eine Kommunikation mit dem EUCHNER Electronic-Key-System (EKS) aus ActiveX-fähigen Programmierumgebungen (z. B. Microsoft Visual Basic®) bzw. Anwenderprogrammen (z. B. Microsoft Excel®) aufzubauen. Hierzu muss das ActiveX®-Modul installiert und in die jeweilige Programmierumgebung eingebunden sein.

1.1 Verwendung des Handbuchs

Dieses Handbuch erläutert die Funktionen des EKS Ethernet ActiveX®-Moduls (Best. Nr. 100 665), ab Version 1.0.3.0.

1.2 Symbolerklärungen

In diesem Handbuch wird zur Visualisierung von wichtigen Hinweisen und nützlichen Informationen folgende Symbolik verwendet:

Information!
Dem Benutzer werden hier wichtige Informationen gegeben.

Achtung!
Gefahr von Datenverlust.

1.3 Voraussetzungen an den Anwender

Für eine sachgerechte Verwendung des EKS Ethernet ActiveX®-Moduls müssen Sie über Vorkenntnisse in der Verwendung von ActiveX®-Modulen verfügen. Für eine problemlose Einbindung der EKS-Hardware in Ihr Gesamtsystem müssen Sie das Handbuch für die Schlüsselaufnahme gelesen und verstanden haben.

1.4 Systemvoraussetzungen

Hardware: Standard-PC, mit Netzwerkanschluss

Software: TCP-IP Protokoll muss installiert sein.

Betriebssystem: Windows® XP
Windows® 7 32-Bit
Windows® 7 64-Bit
Windows® Server 2008 32-Bit
Windows® Server 2008 64-Bit
Windows® Server 2008 R2 64-Bit

2 Supportinformation, Installation und Deinstallation

Um das EUCHNER EKS Ethernet TCP/IP ActiveX®-Modul verwenden zu können, müssen Sie es zuerst installieren. Führen Sie in Abhängigkeit vom vorhandenen Betriebssystem die entsprechende Installationsdatei aus:

- Für Windows® 32-Bit: EKS_Ethernet_ActiveX_Module.msi
- Für Windows® 64-Bit: EKS_Ethernet_ActiveX_Module_x64.msi

Information!
Bei der Installation werden Sie aufgefordert ein Installationsverzeichnis anzugeben. Nach erfolgreicher Installation befindet sich darin:

- das ActiveX®-Modul
- dieses Handbuch im Acrobat PDF-Format

Auf der Installations-CD befinden sich:

- die Programme EKS_Ethernet_ActiveX_Module.msi und EKS_Ethernet_ActiveX_Module_x64.msi
- dieses Handbuch im Acrobat PDF-Format
- Programmierbeispiele für verschiedene Programmierumgebungen

Um das ActiveX®-Modul zu deinstallieren oder um Supportinformationen zu erhalten gehen Sie folgendermaßen vor:

1. Wählen Sie im Betriebssystem *Einstellungen* | *Systemsteuerung* | *Software*.
2. Wählen Sie in der Übersicht der installierten Programme den Eintrag *EUCHNER EKS Ethernet ActiveX Module*. Hier können Sie auch die Supportinformationen anzeigen lassen.

   Information!
   Halten Sie die Supportinformationen bei Rückfragen an EUCHNER immer bereit.

3. Zum Deinstallieren klicken Sie auf *Ändern/Entfernen* und folgen den Anweisungen des Deinstallationsdialogs.

3 Das EKS Ethernet ActiveX®-Modul

3.1 EKS Typelibrary

- Beschreibung EUCHNER EKS Ethernet ActiveX Module
- Bibliothek EKSEthLib
- Dateiname ekseth.ocx
- GUID { 36B62232-F5C4-4b46-BA4A-4B1F3F2C7974 }
- Control EKSETH

3.2 EKS Control

- Control Name EKSETH
- Dateiname ekseth.ocx
- GUID { 72484DED-F77A-487c-9DC7-5751D10DBF17 }
- Methoden 6
- Eigenschaften 13
- Ereignisse 3

Bevor Sie das EKS Ethernet ActiveX®-Modul in Ihrer Applikation verwenden können, müssen Sie Ihrem Projekt die Datei ekseth.ocx hinzufügen. Um eine Applikation zu betreiben, die das ActiveX®-Modul verwendet, müssen Sie es auf Ihrem Rechner installieren.

3.3 Übersicht der Methoden, Eigenschaften und Ereignisse im EKS Ethernet ActiveX®-Modul

Das EKS Ethernet ActiveX®-Modul enthält Methoden, Eigenschaften (Properties) und Ereignisse (Events) die in Ihre Programmierumgebung eingebunden werden können.

- **Methoden** dienen zur Verbindungsaufnahme und zur Datenübertragung zwischen dem Anwenderprogramm und der EKS-Schlüsselaufnahme.
- **Eigenschaften (Properties)** werden für Einstellungen benutzt, geben Zustände wieder und enthalten Daten die vom Schlüssel gelesen werden oder auf den Schlüssel geschrieben werden sollen.
- **Ereignisse (Events)** melden den Abschluss einer Methode oder signalisieren ein Ereignis (z. B. Schlüssel gesteckt).

Alle Methoden, Properties und Events des Objekts EKS werden in folgender Tabelle aufgeführt.

Methoden	**Kapitel**
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
Eigenschaften (Properties)	**Kapitel**
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
Ereignisse (Events)	**Kapitel**
OnKey	3.7.1 OnKey
OnRead	3.7.2 OnRead
OnWrite	3.7.3 OnWrite

3.4 Methoden

3.4.1 Open

- Beschreibung: Öffnet die Verbindung zum EKS mit den eingestellten Properties (*IPAddress, Port*, *KeyType*, *StartAdress*, *CountData* ...).
- Syntax: **Boolean** = *object*.**EKS.Open**;
- Bemerkungen: Das EKS muss angeschlossen und betriebsbereit sein, bevor diese Methode benutzt wird. Die Methode liefert entweder den Rückgabewert *True* (fehlerfreie Ausführung) oder *False* (Statusmeldung wurde erzeugt). Im Statusfall kann über das Property *LastState* die Ursache ermittelt werden. Eine Übersicht der Statusmeldungen für das ActiveX®-Modul finden Sie unter Kapitel 3.5.4. Nach Abschluss der asynchronen Ausführung wird das Event *OnKey* ausgelöst. Um den aktuellen Zustand der Methode *Open* zu erhalten kann das Property *Opening* abgefragt werden. Bei Programmende muss eine geöffnete Verbindung wieder durch Aufruf der Methode *Close* geschlossen werden.

Information!
Die Methode *Open* startet einen Hintergrundprozess der die Kommunikation mit dem Gerät aufbaut. Der Rückgabewert *True* signalisiert nur, dass der Hintergrundprozess gestartet werden konnte. Eine physische Verbindung zum Gerät wird dabei nicht geprüft.

3.4.2 Close

- Beschreibung: Schließt die Verbindung zum EKS.
- Syntax: **Boolean** = *object*.**Close ()**;
- Bemerkungen: Diese Methode muss am Ende des Anwenderprogramms ausgeführt werden, um die Schnittstelle des PCs wieder frei zu geben.

3.4.3 Read

- Beschreibung: Methode um Daten vom Schlüssel zu lesen (Startadresse ist im Property *StartAdress* und die Anzahl der Daten im Property *CountData* festgelegt)
- Syntax: **Boolean** = *object*.**Read ()**;
- Bemerkungen: Liefert die Methode als Rückgabewert *True*, werden die Daten vom EKS gelesen. Diese können nach Auslösen des Events *OnRead* aus dem Property *Data* entnommen werden. Bei Rückgabewert *False* konnte der Leseauftrag nicht fehlerfrei gestartet werden. In diesem Fall befindet sich im Property *LastState* die Statusnummer. Eine Übersicht der Statusmeldungen für das ActiveX®-Modul finden Sie unter Kapitel 3.5.4.

Information!
Wenn Sie nur die Daten des Schlüssels lesen möchten, benötigen Sie keinen expliziten Aufruf der Methode *Read*. Sobald das Event *OnKey* ausgelöst wird und das Property *KeyState* = *EKS_KEY_IN* ist, stehen die Daten des aktuellen Schlüssels im Property *Data* zur Verfügung. Vor dem Auslösen des Events *OnKey* wird intern im ActiveX®-Modul die Methode *Read* ausgeführt.

3.4.4 Write

- Beschreibung: Methode um Daten auf den Schlüssel zu schreiben (Startadresse ist im Property *StartAdress* und die Anzahl der Daten im Property *CountData* festgelegt)
- Syntax: **Boolean** = *object*.**Write ()**;
- Bemerkungen: Liefert die Methode als Rückgabewert *True*, werden die Daten auf den Schlüssel geschrieben. Der Schreibauftrag ist nach Auslösen des Events *OnWrite* abgeschlossen. Bei Rückgabewert *False* konnte der Schreibauftrag nicht fehlerfrei gestartet werden. In diesem Fall befindet sich im Property *LastState* die Statusnummer. Eine Übersicht der Statusmeldungen für das ActiveX®-Modul finden Sie unter Kapitel 3.5.4

3.4.5 getData

- Beschreibung: Lesender Zugriff auf internen Speicherbereich des ActiveX®-Moduls, in dem die gelesenen Daten der Methode *Read* oder des Events *OnKey* abgelegt werden.
- Syntax: **short** = *object*.**getData** (*short* ByteIndex);
- Bemerkungen: Mit der Methode *getData* kann der interne Speicherbereich des ActiveX®-Moduls gelesen werden. Nach Auslösen der Events *OnRead* oder *OnKey* stehen die Daten des Schlüssels im internen Speicher zur Verfügung und können mit Hilfe von *getData* ausgelesen werden. In den Properties *StartAdress* und *CountData* wird der Bereich festgelegt, ab welchem Byte gelesen (Methode *Read*) werden soll.

Information!
Es handelt sich hier um eine zusätzliche Möglichkeit auf den internen Speicher des ActiveX®-Moduls zuzugreifen. Diese Methode kann in Programmiersprachen verwendet werden, welche keine Arrays unterstützten. Üblicherweise wird auf den internen Speicherbereich über das Property *Data,* siehe Kapitel 3.5.12, zugegriffen.

3.4.6 setData

- Beschreibung: Schreibender Zugriff auf internen Speicherbereich des ActiveX®-Moduls, in dem die zu schreibenden Daten der Methode *Write* abgelegt werden.
- Syntax: *object*.**setData** (*short* ByteIndex, *short* DataValue);
- Bemerkungen: Mit der Methode *setData* kann in den internen Speicherbereich des ActiveX®-Moduls geschrieben werden. Nach Auslösen des Events *OnWrite* werden die Daten aus dem Zwischenspeicher auf den Schlüssel geschrieben. In den Properties *StartAdress* und *CountData* wird der Bereich festgelegt, ab welchem Byte geschrieben (Methode *Write*) werden soll.

Information!
Es handelt sich hier um eine zusätzliche Möglichkeit auf den internen Speicher des ActiveX®-Moduls zuzugreifen. Diese Methode kann in Programmiersprachen verwendet werden, welche keine Arrays unterstützten. Üblicherweise wird auf den internen Speicherbereich über das Property *Data,* siehe Kapitel 3.5.12, zugegriffen.

3.5 Eigenschaften (Properties)

3.5.1 Port

- Beschreibung: Wählt den Port der TCP-Verbindung
- Syntax: *object*.**Port = String** Value;
- Bemerkungen: Mögliche Werte sind:
  2444
  2445
  …
  Dieses Property wird durch Aufruf der Methode *Open* übernommen. Der Wert muss mit der Einstellung am EKS übereinstimmen
- Datentyp: **String**
- Standardwert: 2444

3.5.2 IPAddress

- Beschreibung: Wählt die IP-Adresse der TCP-Verbindung
- Syntax: *object*.**IPAddress = String** Value;
- Bemerkungen: Mögliche Werte sind:
  192.168.1.1
  …
  Dieses Property wird durch Aufruf der Methode *Open* übernommen. Der Wert muss mit der Einstellung am EKS übereinstimmen
- Datentyp: **String**
- Standardwert: 192.168.1.1

3.5.3 KeyType

- Beschreibung: Legt den verwendeten Schlüsseltyp fest (Nur-Lese-Schlüssel* oder Schreib-/Lese-Schlüssel)
  *Früherer Transpondertyp. Wir empfehlen diesen Transpondertyp bei Neuinstallationen nicht mehr einzusetzen.
- Syntax: *object*.**KeyType = KeyTypeConstants** Value;
- Bemerkungen: Mögliche Werte sind:
  EKS_KEY_READWRITE = 1
  EKS_KEY_READONLY = 8
  Dieses Property wird durch Aufruf der Methode *Open* übernommen.
- Datentyp: **KeyTypeConstants** (Enumeration)
- Standardwert: EKS_KEY_READWRITE

3.5.4 LastState (ReadOnly)

- Beschreibung: Status der zuletzt ausgeführten Methode (0=OK oder Statusnummer)
- Syntax: **long =** *object*.**LastState**;
- Bemerkungen: Nach Ausführen einer Methode (*Read*, *Write, …*) oder eines Events (*OnKey*, *OnRead*, ...) kann hier festgestellt werden, ob die Methode fehlerfrei durchgeführt wurde. Statusnummern im Bereich von 0 bis 127 (0_{hex} bis $7F_{hex}$) werden vom EKS generiert und sind im Handbuch der EKS-Schlüsselaufnahme dokumentiert. Statusnummern zwischen 128 und 255 (80_{hex} bis FF_{hex}) generiert das ActiveX®-Modul.
- Datentyp: **long**
- Liste der Statusnummern des ActiveX®-Moduls:

Achtung!
Sofort nachdem eine Methode aufgerufen oder ein Event ausgelöst wurde, sollten Sie den Wert im Property *LastState* abfragen. Das Property *LastState* könnte sonst von einer anderen Methode überschrieben werden, da immer nur die Statusmeldung der zuletzt ausgeführten Methode im Property *LastState* steht. Das gilt auch bei internen Methoden, die im Hintergrund laufen und nicht von Ihnen gestartet wurden.

Wert hex	Wert dec	Beschreibung
0x90	144	**WrongParam** Der angegebene TCP-Port im Property *Port* liegt nicht im Bereich >1024 und <65535
0xA0	160	**DeviceNotOpened** Die Verbindung zum EKS wurde nicht geöffnet, bitte führen Sie die Methode *Open* aus.
0xB0	176	**ReadTimeOut** Die Methode *Read* konnte nicht korrekt abgeschlossen werden, die Timeout-Zeit wurde überschritten.
0xB1	177	**WriteTimeOut** Die Methode *Write* konnte nicht korrekt abgeschlossen werden, die Timeout-Zeit wurde überschritten.
0xB2	178	**TimeOut** Bei der Behandlung einer internen Methode des ActiveX®-Moduls wurde die Timeout-Zeit überschritten.
0xC0	192	**NothingToRead** Die Anzahl der zu lesenden Daten, definiert von dem Property *CountData,* ist *0*.
0xC1	193	**NothingToWrite** Die Anzahl der zu schreibenden Daten, definiert von dem Property *CountData,* ist *0*.
0xE0	224	**OpenFailed** Die Methode *Open* ist fehlgeschlagen.
0xE1	225	**OpenActive** Die Methode *Open* ist noch aktiv.
0xE8	232	**Suspend** Der Computer wird in den Suspendmodus versetzt.
0xE9	233	**ResumeSuspend** Der Suspendmodus wurde beendet.
0xEA	234	**ConnectionTimeOut** Verbindungs-Timeout zum EKS. Es konnte in der vorgegebenen Zeit keine Verbindung zum EKS hergestellt werden.
0xEB	235	**ConnectionLost** Die Verbindung zum EKS wurde unterbrochen.
0xEC	236	**Reconnect** Die Verbindung zum EKS wurde wieder hergestellt.
0xFF	255	**Busy** Das ActiveX®-Modul ist mit der Abarbeitung einer Methode beschäftigt, die Anforderung kann nicht ausgeführt werden.

3.5.5 StartAdress

- Beschreibung: Die Startadresse des Speicherbereichs auf dem Schlüssel, von der ab gelesen werden soll (*Read*) bzw. ab der geschrieben werden soll (*Write*).
- Syntax: *object*.**StartAdress = short** Value;
- Bemerkungen: Legt die Startadresse der zu lesenden Daten bei der Methode *Read* sowie die Startadresse der zu schreibenden Daten bei der Methode *Write* fest. Die gelesenen Daten befinden sich nach erfolgreicher Lesemethode im Property *Data*. Dort müssen auch die zu schreibenden Daten hinterlegt werden. Das Property *StartAdress* muss vor dem Aufruf der Methoden gesetzt werden, um die Startadresse beim nächsten Aufruf verwenden zu können.

Information!
Beim Schreib-/Lese-Schlüssel mit frei programmierbaren 116 Bytes ist der Speicher in 4-Byte-Blöcken organisiert. Dies bedeutet, die Start-Adresse muss beim Schreiben im Bereich Byte Nr. 0 bis Byte Nr. 112, immer in 4-Byte-Schritten, angegeben werden (Byte Nr. 0, 4, 8 ... 112). Außerdem muss immer in einem Vielfachen von 4-Bytes großen Blöcken geschrieben werden (4, 8, 12 ... 116 Bytes)!

Beim Lesen kann allerdings wiederum byteweise auf den Speicher zugegriffen werden, ohne die oben genannte Einschränkung beim Schreiben.

Der Schreib-/Lese-Schlüssel hat zusätzlich eine einmalige 8-Bytes große Serien-Nummer, die bei der Schlüssel-Produktion per Laser eingeschrieben wird und somit absolut unzerstörbar gespeichert ist. Die Serien-Nummer kann daher nicht geändert werden. Diese Serien-Nummer dient zur sicheren Unterscheidung eines jeden einzelnen Schlüssels. Für eine sichere Unterscheidung ist es erforderlich alle 8 Bytes komplett auszuwerten. Die Serien-Nummer schließt sich an den frei programmierbaren Speicher an. Die Serien-Nummer kann unter Eingabe der Start-Adresse Byte Nr. 116 und Anzahl Bytes 8 ausgelesen werden.

- Datentyp: **short**
- Standardwert: 0

3.5.6 CountData

- Beschreibung: Die Anzahl der zu schreibenden bzw. zu lesenden Daten.
- Syntax: *object*.**CountData = short** Value;
- Bemerkungen: Legt die Anzahl der zu lesenden Daten bei den Methoden *Read* und die Anzahl der zu schreibenden Daten bei der Methode *Write* fest. Die gelesenen Daten befinden sich nach erfolgreicher Lesemethode im Property *Data*. Dort müssen auch die zu schreibenden Daten hinterlegt werden. Das Property *CountData* muss vor dem Aufruf der Methoden gesetzt werden, um die Anzahl der zu lesenden/schreibenden Daten beim nächsten Aufruf verwenden zu können.

Information!
Beim Schreib-/Lese-Schlüssel mit frei programmierbaren 116 Bytes ist der Speicher in 4-Byte-Blöcken organisiert. Dies bedeutet, die Start-Adresse muss beim Schreiben im Bereich Byte Nr. 0 bis Byte Nr. 112, immer in 4-Byte-Schritten, angegeben werden (Byte Nr. 0, 4, 8 ... 112). Außerdem muss immer in einem Vielfachen von 4-Bytes großen Blöcken geschrieben werden (4, 8, 12 ... 116 Bytes)!

Beim Lesen kann allerdings wiederum byteweise auf den Speicher zugegriffen werden, ohne die oben genannte Einschränkung beim Schreiben.

Der Schreib-/Lese-Schlüssel hat zusätzlich eine einmalige 8-Bytes große Serien-Nummer, die bei der Schlüssel-Produktion per Laser eingeschrieben wird und somit absolut unzerstörbar gespeichert ist. Die Serien-Nummer kann daher nicht geändert werden. Diese Serien-Nummer dient zur sicheren Unterscheidung eines jeden einzelnen Schlüssels. Für eine sichere Unterscheidung ist es erforderlich alle 8 Bytes komplett auszuwerten. Die Serien-Nummer schließt sich an den frei programmierbaren Speicher an. Die Serien-Nummer kann unter Eingabe der Start-Adresse Byte Nr. 116 und Anzahl Bytes 8 ausgelesen werden.

- Datentyp: **short**
- Standardwert: 4

3.5.7 Opening

- Beschreibung: Zustand der Methode *Open*
- Syntax: **bool =** *object*.**Opening**;
- Bemerkungen: Gibt das Property *Opening* den Wert *True* zurück, ist gerade die Methode *Open* aktiv. Solange kann keine weitere Methode aufgerufen werden
- Datentyp: **bool**
- Standardwert: false

3.5.8 Reading

- Beschreibung: Zustand der Methode *Read*
- Syntax: **bool =** *object*.**Reading**;
- Bemerkungen: Gibt das Property *Reading* den Wert *True* zurück, ist gerade die Methode *Read* aktiv. Die Daten des Schlüssels stehen noch nicht im Property *Data* zur Verfügung. Solange kann keine weitere Methode aufgerufen werden.
- Datentyp: **bool**
- Standardwert: false

3.5.9 Writing

- Beschreibung: Zustand der Methode *Write*
- Syntax: **bool =** *object*.**Writing**;
- Bemerkungen: Gibt das Property *Writing* den Wert *True* zurück, ist gerade die Methode *Write* aktiv. Der Schreibauftrag ist noch aktiv und die Daten wurden noch nicht vollständig auf den Schlüssel geschrieben. Solange kann keine weitere Methode aufgerufen werden.
- Datentyp: **bool**
- Standardwert: false

3.5.10 KeyState

- Beschreibung: Gibt den Status des letzten Events zurück.
- Syntax: **bool =** *object*.**KeyState**;
- Bemerkungen: Mögliche Parameter sind:
  - EKS_KEY_IN = 1
  - EKS_KEY_OUT = 2
  - EKS_KEY_OTHER = 3
- Datentyp: **KeyStateConstants** (Enumeration)
- Standardwert: EKS_KEY_OUT

3.5.11 Version

- Beschreibung: Gibt die aktuelle Version des EKS Ethernet ActiveX®-Moduls zurück
- Syntax: **String** Value = *object*.**Version**;
- Datentyp: **String**

3.5.12 Data

- Beschreibung: Speicherbereich, in dem gelesene Daten der Methoden *Read* oder des Events *OnKey* oder zu schreibende Daten der Methode *Write* abgelegt werden.
- Syntax: **short =** *object*.**Data** (**short** ByteIndex);
- Bemerkungen: Das Property *Data* stellt einen Zwischenspeicher für alle Daten dar, die vom Schlüssel gelesen werden und auf den Schlüssel geschrieben werden sollen. Die Daten des Schlüssels werden Byte-weise bereitgestellt bzw. zugewiesen. Nach Auslösen der Events *OnRead* oder *OnKey* stehen die Daten des Schlüssels im Property *Data* zur Verfügung. Nach Auslösen des Events *OnWrite* wurden die Daten aus dem Property *Data* auf den Schlüssel geschrieben. In den Properties *StartAdress* und *CountData* wird der Bereich festgelegt, ab welchem Byte gelesen (Methode *Read*) bzw. geschrieben (Methode *Write*) werden soll.
- Datentyp: **short**
- Standardwert: -12851 ($CDCD_{hex}$)
  Der Standardwert ist verfügbar, wenn noch keine Daten vom Schlüssel gelesen wurden oder sich kein Schlüssel in der Schlüsselaufnahme befindet.

3.5.13 Debug

- Beschreibung: Ist das Property *Debug* auf den Wert *true* gesetzt, wird beim Beenden der Debug-Sitzung einer Programmierumgebung für alle Instanzen die Methode *Close* aufgerufen.
- Syntax: **bool =** *object*.**Debug**;
- Bemerkungen: Das ActiveX®-Modul wird von verschiedenen Programmierumgebungen beim Beenden der Debug-Sitzung nicht korrekt zerstört. Das Property *Debug* muss z. B. in Microsoft Visual Basic® und Microsoft Excel® zur Entwicklung der Anwendung auf den Wert *true* gesetzt werden, damit beim Beenden der Debug-Sitzung, ohne den expliziten Aufruf der Methode *Close*, die TCP-Verbindung geschlossen wird. Wird eine Debug-Sitzung beendet, bevor die Methode *Close* aufgerufen wird, bleibt die Verbindung zu EKS geöffnet. Hat das Property *Debug* den Wert *true*, dann werden die TCP-Verbindungen **aller** ActiveX®-Instanzen beim Beenden der Debug-Sitzung geschlossen.
- Datentyp: **bool**
- Standardwert: false

Information!
Bitte beachten Sie, dass Sie das Property *Debug* nur in der Debug-Sitzung auf den Wert *true* setzen. Wird bei der Verwendung mehrerer Instanzen des ActiveX®-Moduls **ein** Control zerstört, so wird bei allen anderen Instanzen die Verbindung zum EKS geschlossen.

3.6 Konstanten

Dieser Abschnitt zeigt alle Konstanten, die in den Properties des EKS Ethernet ActiveX®-Moduls verwendet werden. Die Konstanten werden auch in der Beschreibung der Eigenschaften und Methoden aufgeführt, in denen sie verwendet werden.

***KeyStateConstants* (verwendet im Property *KeyState*)**

Wert	Konstante
1	EKS_KEY_IN
2	EKS_KEY_OUT
3	EKS_KEY_OTHER

***KeyTypeConstants* (verwendet im Property *KeyType*)**

Wert	Konstante
1	EKS_KEY_READWRITE
8	EKS_KEY_READONLY

3.7 Ereignisse (Events)

3.7.1 OnKey

- Beschreibung: Dieses Event muss im Anwenderprogramm definiert sein und wird vom ActiveX®-Modul aufgerufen.
- Syntax: **Private Sub** *object*_OnKey ()
- Bemerkungen: Um dieses Event zu nutzen, muss im Anwenderprogramm eine Methode mit dem Namen *OnKey* definiert werden, welche vom ActiveX®-Modul aufgerufen wird, sobald sich eine Veränderung am EKS (Schlüssel gesteckt/Schlüssel gezogen, usw.) ergibt. Im Anwenderprogramm kann der Benutzer dann abfragen, welches Event aufgetreten ist (EKS_KEY_IN, EKS_KEY_OUT, EKS_KEY_OTHER). Das Event *OnKey* mit dem Property *KeyState=EKS_EVENT_KEYIN* wird ausgelöst, wenn sich ein neuer Schlüssel im EKS befindet. Der Anwender kann dann die Daten des Schlüssels aus dem Property *Data* auslesen **ohne einen Aufruf der Methode *Read***. Das Event *OnKey* mit dem Property *KeyState=EKS_KEY_OUT* wird beim Entfernen des Schlüssels ausgelöst. Das Event *OnKey* mit dem Property *KeyState=EKS_KEY_OTHER* wird ausgelöst, wenn das ActiveX®-Modul einen Status feststellt. Die zugehörige Statusnummer kann im Property *LastState* ausgelesen werden.

Information!
Die Erkennung, ob ein Schlüssel gesteckt oder gezogen wurde, wird über ein Netzwerktelegramm gemeldet (siehe Handbuch für Schlüsselaufnahme EKS Ethernet TCP/IP). Das Telegramm wird sofort nach dem Stecken bzw. Ziehen des Schlüssels von der Schlüsselaufnahme an den aktuell verbundenen ActiveX®-Client gesendet.

3.7.2 OnRead

- Beschreibung: Dieses Event muss im Anwenderprogramm definiert sein und wird vom ActiveX®-Modul aufgerufen.
- Syntax: **Private Sub** *object*_**OnRead ()**
- Bemerkungen: Um dieses Event zu nutzen, muss im Anwenderprogramm eine Methode mit dem Namen *OnRead* definiert werden, welche vom ActiveX®-Modul aufgerufen wird, sobald im ActiveX®-Modul die Methode *Read* abgeschlossen ist. Die zugehörige Statusnummer kann im Property *LastState* ausgelesen werden.

3.7.3 OnWrite

- Beschreibung: Dieses Event muss im Anwenderprogramm definiert sein und wird vom ActiveX®-Modul aufgerufen.
- Syntax: **Private Sub** *object*_**OnWrite ()**
- Bemerkungen: Um dieses Event zu nutzen, muss im Anwenderprogramm eine Methode mit dem Namen *OnWrite* definiert werden, welche vom ActiveX®-Modul aufgerufen wird, sobald die Metode *Write* im ActiveX®-Modul abgeschlossen ist. Die zugehörige Statusnummer kann im Property *LastState* ausgelesen werden.

4 Beispiele

Information!
Auf der Installations-CD finden Sie Beispiele zur Einbindung des EKS Ethernet ActiveX®-Moduls in verschiedene Programmierumgebungen.

4.1 Verbindungsaufbau mit der EKS-Schlüsselaufnahme

Das folgende Beispiel zeigt wie die Methode *Open* verwendet werden kann. Die dargestellten Werte entsprechen den Standardeinstellungen der Properties. Für Ihre Anwendung kann es notwendig sein diese Werte zu verändern.

1. Gewünschte Werte in den Properties einstellen. Dies kann auch im Programmiertool (z.B. Visual Basic®) über die Eigenschaften des Objekts *EKS* erfolgen:

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

2. Die gewünschten Schreib-/Lese-Parameter einstellen (kann auch nach Öffnen der Schnittstelle erfolgen):

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

3. Schnittstelle öffnen:

```
EKS.Open
```

Information!
Wenn mit den gezeigten Standardwerten gearbeitet wird, genügt als einzige Zeile der Aufruf *EKS.Open.*

4.2 Beispiel eines Event-Aufrufs in Visual Basic®

```
Private Sub EKS_OnKey( )
      Select Case KeyState
            Case EKS_EVENT_KEYIN
                  Anwender Funktionen KeyIn
                  ' z. B. Schlüsseldaten des EKS Key auslesen
                  ' ACHTUNG ! Kein Aufruf der Read Methode erforderlich !
                  for i=0 to 123
                        SchluesselDaten = SchluesselDaten & EKS.Data(i)
                  Next i
            Case EKS_EVENT_KEYOUT
                  Anwender Funktionen KeyOut
                  ' z. B. Schlüsseldaten der Anwendersoftware löschen
                  SchluesselDaten = "-"
            Case EKS_EVENT_OTHER
                  Anwender Funktionen Andere
                  ' z. B. Statusnummer abfragen
                  Statusnummer = EKS.LastState
            End Select
      End Sub
```

More than safety.

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

Telefon 0711 / 75 97 - 0
Telefax 0711 / 75 33 16
www.euchner.de · info@euchner.de

EUCHNER