

**Electronic-Key-System**  
**Handbuch**  
**Software ActiveX®-Modul seriell / USB**  
**Best. Nr. 2098655**



**EKS.**



**EUCHNER**

More than safety.

## Inhaltsverzeichnis

<b>1 Allgemeine Hinweise .....</b>	<b>4</b>
1.1 Verwendung des Handbuchs .....	4
1.2 Symbolerklärungen .....	4
1.3 Voraussetzungen an den Anwender .....	4
1.4 Systemvoraussetzungen .....	4
<b>2 Supportinformation, Installation und Deinstallation .....</b>	<b>5</b>
<b>3 Das EKS ActiveX®-Modul .....</b>	<b>6</b>
3.1 EKS Typelibrary .....	6
3.2 EKS Control .....	6
3.3 Übersicht der Methoden, Eigenschaften und Ereignisse im EKS ActiveX®-Modul .....	7
3.4 Methoden .....	8
3.4.1 Open .....	8
3.4.2 Close .....	8
3.4.3 Read .....	8
3.4.4 Write .....	9
3.4.5 getData .....	9
3.4.6 setData .....	9
3.5 Eigenschaften (Properties) .....	10
3.5.1 BaudRate .....	10
3.5.2 Port .....	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 BlockSize .....	13
3.5.8 PollingTime .....	13
3.5.9 Opening .....	13
3.5.10 Reading .....	13
3.5.11 Writing .....	14
3.5.12 KeyState .....	14
3.5.13 Version .....	14
3.5.14 Data .....	14
3.5.15 Debug .....	15
3.6 Konstanten .....	15

3.7 Ereignisse (Events).....16

    3.7.1 OnKey .....16

    3.7.2 OnRead .....16

    3.7.3 OnWrite .....16

**4 Beispiele .....17**

    4.1 Verbindungsaufbau mit der EKS-Schlüsselaufnahme.....17

    4.2 Beispiel eines Event-Aufrufs in Visual Basic®.....18

# 1 Allgemeine Hinweise

Die Anbindung der Electronic-Key-System (EKS) Schlüsselaufnahme mit serieller und USB-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 nach dem 3964R Übertragungsprotokoll ab. Das ActiveX®-Modul dient dabei als Protokolltreiber.

Mit Hilfe des EKS 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 ActiveX®-Moduls (Best. Nr. 8098708), ab Version 1.0.3.0. Das Handbuch gilt nicht für frühere Softwarestände oder die Vorgängerversion des EKS ActiveX®-Moduls (Best. Nr. 084708).

## 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 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

<b>Hardware:</b>	Standard-PC, keine besonderen Anforderungen
<b>Software:</b>	Wenn Sie eine EKS-Schlüsselaufnahme mit USB-Schnittstelle verwenden, muss der EKS-USB-Treiber (Best. Nr. 094 376) ab Version 1.3.0.0 auf dem System installiert sein.
<b>Betriebssystem:</b>	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 Supportinformation, Installation und Deinstallation

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

- ▶ Für 32-Bit Anwenderprogramm: EKS\_ActiveX\_Module.msi
- ▶ Für 64-Bit Anwenderprogramm: EKS\_ActiveX\_Module\_x64.msi



### Information!

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

- ▶ das ActiveX®-Modul

Im heruntergeladenen zip-Archiv befinden sich:

- ▶ die Installationsdateien EKS\_ActiveX\_Module.msi und EKS\_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 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 ActiveX®-Modul

### 3.1 EKS Typelibrary

- ▶ Beschreibung EUCHNER EKS ActiveX Module
- ▶ Bibliothek EKSLib
- ▶ Dateiname eks.ocx
- ▶ GUID { 62A51CD4-76C1-453D-B258-804D12988851 }
- ▶ Control EKS

### 3.2 EKS Control

- ▶ Control Name EKS
- ▶ Dateiname eks.ocx
- ▶ GUID { 64CAE8A8-3CB8-4929-A90F-57499A6E83F3 }
- ▶ Eigenschaften 15
- ▶ Ereignisse 3
- ▶ Methoden 4

Bevor Sie das EKS ActiveX®-Modul in Ihrer Applikation verwenden können, müssen Sie Ihrem Projekt die Datei eks.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 ActiveX®-Modul

Das EKS ActiveX®-Modul enthält Methoden, Eigenschaften (Properties) und Ereignisse (Events) die in Ihre Programmierungsumgebung 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	<a href="#">3.4.1 Open</a>
Close	<a href="#">3.4.2 Close</a>
Read	<a href="#">3.4.3 Read</a>
Write	<a href="#">3.4.4 Write</a>
getData	<a href="#">3.4.5 getData</a>
setData	<a href="#">3.4.6 setData</a>
Eigenschaften (Properties)	Kapitel
BaudRate	<a href="#">3.5.1 BaudRate</a>
Port	<a href="#">3.5.2 Port</a>
KeyType	<a href="#">3.5.3 KeyType</a>
LastState	<a href="#">3.5.4 LastState (ReadOnly)</a>
StartAdress	<a href="#">3.5.5 StartAdress</a>
CountData	<a href="#">3.5.6 CountData</a>
BlockSize	<a href="#">3.5.7 BlockSize</a>
PollingTime	<a href="#">3.5.8 PollingTime</a>
Opening	<a href="#">3.5.9 Opening</a>
Reading	<a href="#">3.5.10 Reading</a>
Writing	<a href="#">3.5.11 Writing</a>
KeyState	<a href="#">3.5.12 KeyState</a>
Version	<a href="#">3.5.13 Version</a>
Data	<a href="#">3.5.14 Data</a>
Debug	<a href="#">3.5.15 Debug</a>
Ereignisse (Events)	Kapitel
OnKey	<a href="#">3.7.1 OnKey</a>
OnRead	<a href="#">3.7.2 OnRead</a>
OnWrite	<a href="#">3.7.3 OnWrite</a>

## 3.4 Methoden

### 3.4.1 Open

- ▶ Beschreibung Öffnet die serielle Schnittstelle zum EKS mit den eingestellten Properties (*BaudRate*, *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 serielle 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 serielle Schnittstelle zum EKS.
- ▶ Syntax **Boolean** = *object*.**Close** ();
- ▶ Bemerkungen Diese Methode muss am Ende des Anwenderprogramms ausgeführt werden, um die serielle 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 unterstützen. Üblicherweise wird auf den internen Speicherbereich über das Property *Data*, siehe Kapitel 3.5.14, 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 unterstützen. Üblicherweise wird auf den internen Speicherbereich über das Property *Data*, siehe Kapitel 3.5.14, zugegriffen.

## 3.5 Eigenschaften (Properties)

### 3.5.1 BaudRate

- ▶ Beschreibung Baudrate einstellen
- ▶ Syntax *object.BaudRate = BaudRateConstants* Value;
- ▶ Bemerkungen Mit dieser Eigenschaft muss dieselbe Baudrate eingestellt werden, wie sie auch über die DIP-Schalter am EKS ausgewählt wurde. Mögliche Werte sind:  
EKS\_BAUD\_9600 = 9600  
EKS\_BAUD\_28800 = 28800  
Dieses Property wird durch Aufruf der Methode *Open* übernommen.
- ▶ Datentyp **BaudRateConstants** (Enumeration)
- ▶ Standardwert EKS\_BAUD\_9600

### 3.5.2 Port

- ▶ Beschreibung Wählt die serielle Schnittstelle am PC
- ▶ Syntax *object.Port = String* Value;
- ▶ Bemerkungen Mögliche Werte sind:  
COM1  
COM2  
...  
Dieses Property wird durch Aufruf der Methode *Open* übernommen.
- ▶ Datentyp **String**
- ▶ Standardwert COM1

### 3.5.3 KeyType

- ▶ Beschreibung Legt den verwendeten Schlüsseltyp fest. Im EKS wird nur noch der Schreib-/Leseschlüssel unterstützt.
- ▶ Syntax *object.KeyType = KeyTypeConstants* Value;
- ▶ Bemerkungen Wert ist:  
EKS\_KEY\_READWRITE = 1  
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<sub>hex</sub> bis 7F<sub>hex</sub>) werden vom EKS generiert und sind im Handbuch der EKS-Schlüsselaufnahme dokumentiert. Statusnummern zwischen 128 und 255 (80<sub>hex</sub> bis FF<sub>hex</sub>) 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		Beschreibung
hex	dec	
0x88	136	<b>Timeout</b> Die Timeoutzeit von ca. 2 s beim 3964R Protokoll nach dem Senden des STX Kommandos wurde überschritten, das Protokoll wird bis zu 6 x wiederholt.
0x89	137	<b>NAKReceived</b> NAK (15 <sub>hex</sub> ) vom EKS empfangen -> Protokollfehler
0x8A	138	<b>Collision</b> Kollision beim 3964R Protokoll.
0x8B	139	<b>WrongBaudrate</b> Vermutlich ist die falsche Baudrate am EKS oder im ActiveX®-Modul angewählt.
0xA0	160	<b>DeviceNotOpened</b> Die Verbindung zum EKS wurde nicht geöffnet, bitte führen Sie die Methode <i>Open</i> aus.
0xA1	161	<b>DeviceNotAvailable</b> Die gewählte serielle Schnittstelle ist nicht verfügbar.
0xA2	162	<b>DeviceInUse</b> Die gewählte serielle Schnittstelle wird von einer anderen Anwendung verwendet und steht nicht zur Verfügung.
0xB0	176	<b>ReadTimeout</b> Die Methode <i>Read</i> konnte nicht korrekt abgeschlossen werden, die Timeout-Zeit wurde überschritten.
0xB1	177	<b>WriteTimeout</b> Die Methode <i>Write</i> konnte nicht korrekt abgeschlossen werden, die Timeout-Zeit wurde überschritten.
0xB2	178	<b>TimeOut</b> Bei der Behandlung einer internen Methode des ActiveX®-Moduls wurde die Timeout-Zeit überschritten.
0xC0	192	<b>NothingToRead</b> Die Anzahl der zu lesenden Daten, definiert von dem Property <i>CountData</i> , ist 0.
0xC1	193	<b>NothingToWrite</b> Die Anzahl der zu schreibenden Daten, definiert von dem Property <i>CountData</i> , ist 0.
0xD0	208	<b>PortNotOpened</b> Die serielle Schnittstelle ist nicht geöffnet, bitte überprüfen Sie die Einstellung des Property <i>Port</i> und führen Sie die Methode <i>Open</i> aus.
0xE0	224	<b>OpenFailed</b> Die Methode <i>Open</i> ist fehlgeschlagen.
0xE1	225	<b>OpenActive</b> Die Methode <i>Open</i> ist noch aktiv.
0xE6	230	<b>USBReConnected</b> Die Verbindung zum EKS USB wurde wieder hergestellt.
0xE7	231	<b>USBDisConnected</b> Die Verbindung zum EKS USB wurde getrennt.
0xE8	232	<b>Suspend</b> Der Computer wird in den Suspendmodus versetzt.
0xE9	233	<b>ResumeSuspend</b> Der Suspendmodus wurde beendet.
0xFF	255	<b>Busy</b> Das ActiveX®-Modul ist mit der Abarbeitung einer Methode beschäftigt, die Anforderung kann nicht ausgeführt werden.

### 3.5.5 StartAddress

- ▶ Beschreibung Die Startadresse des Speicherbereichs auf dem Schlüssel, von der ab gelesen werden soll (*Read*) bzw. ab der geschrieben werden soll (*Write*).
- ▶ Syntax *object.StartAddress* = **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 *StartAddress* 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 Byte-weise auf den Speicher zugegriffen werden, ohne die oben genannte Einschränkung beim Schreiben.

Der Schreib-/Lese-Schlüssel hat zusätzlich eine einmalige 8-Byte große Seriennummer, die bei der Schlüssel-Produktion absolut unzerstörbar in den Speicher geschrieben wird. Die Seriennummer kann daher nicht geändert werden. Diese Seriennummer dient zur sicheren Unterscheidung eines jeden einzelnen Schlüssels. Für eine sichere Unterscheidung ist es erforderlich, alle 8 Bytes komplett auszuwerten. Die Seriennummer schließt sich an den frei programmierbaren Speicher an. Die Seriennummer 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 Byte-weise auf den Speicher zugegriffen werden, ohne die oben genannte Einschränkung beim Schreiben.

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

- ▶ Datentyp **short**
- ▶ Standardwert 4

### 3.5.7 BlockSize

- ▶ Beschreibung Legt die Blockgröße der Datenübertragung fest.
- ▶ Syntax *object.BlockSize* = **short** Value;
- ▶ Bemerkungen Definiert die Blockgröße der Datenpakete im 3964R-Protokoll. Für dieses Property kann der Standardwert beibehalten werden, wenn keine älteren Geräte der EKS Baureihe genutzt werden.
- ▶ Datentyp **short**
- ▶ Standardwert 124

### 3.5.8 PollingTime

- ▶ Beschreibung Stellt die Zeit [ms] ein, nach der das ActiveX®-Modul beim EKS den Schlüsselstatus abfragt.
- ▶ Syntax *object.PollingTime* = **short** Value;
- ▶ Bemerkungen Über dieses Property kann das Event-Verhalten des ActiveX®-Moduls beeinflusst werden. Ist im Property PollingTime eine Abfragezeit = 0 ms eingestellt, ist das Polling ausgeschaltet.
- ▶ Datentyp **short**
- ▶ Standardwert 0



#### Information!

Im Normalfall ist ein Polling nicht erforderlich, da die Transpondererkennung über das CTS-Signal erfolgt. Polling ist nur sinnvoll, wenn kein CTS-Signal verfügbar ist.

Die Pollingzeit sollte nicht zu niedrig gesetzt werden (< 500 ms), da sonst für weitere Methodenaufrufe keine Zeit zur Verfügung steht. Dies kann zu einem häufigen *LastState* von *0xFF* führen.

### 3.5.9 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

### 3.5.10 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

### 3.5.11 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**

### 3.5.12 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**

### 3.5.13 Version

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

### 3.5.14 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<sub>hex</sub>)**  
Der Standardwert ist verfügbar, wenn noch keine Daten vom Schlüssel gelesen wurden oder sich kein Schlüssel in der Schlüsselaufnahme befindet.

### 3.5.15 Debug

- ▶ Beschreibung Ist das Property *Debug* auf den Wert *true* gesetzt, wird beim Beenden der Debug-Sitzung einer Programmierungsumgebung der COM-Port geschlossen.
- ▶ Syntax **bool = object.Debug;**
- ▶ Bemerkungen Das ActiveX®-Modul wird von verschiedenen Programmierungsumgebungen 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*, der COM-Port geschlossen wird. Wird eine Debug-Sitzung beendet, bevor die Methode *Close* aufgerufen wird, so erhalten Sie bei der nächsten Debug-Sitzung einen *LastState* von 162 (*DeviceInUse*). Hat das Property *Debug* den Wert *true*, dann werden die COM-Ports **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 der COM-Port geschlossen.

## 3.6 Konstanten

Dieser Abschnitt zeigt alle Konstanten, die in den Properties des EKS 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

#### BaudRateConstants (verwendet im Property *BaudRate*)

Wert	Konstante
9600	EKS_BAUD_9600
28800	EKS_BAUD_28800

## 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, erfolgt über den Zustand der CTS Leitung (siehe Handbuch für EKS Schlüsselaufnahme).

### 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 Methode *Write* im ActiveX®-Modul abgeschlossen ist. Die zugehörige Statusnummer kann im Property *LastState* ausgelesen werden.

## 4 Beispiele

**Information!**

Im zip-Archiv finden Sie Beispiele zur Einbindung des EKS 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 = "COM1"

EKS.BaudRate = EKS\_BAUD\_9600

EKS.KeyType = EKS\_KEY\_READWRITE

EKS.PollingTime = 0

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

EKS.StartAdress = 0

EKS.CountData = 4

3. Serielle 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
```



Microsoft Windows® und ActiveX® sind  
eingetragene Warenzeichen der  
Microsoft Corporation

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

Telefon 0711 / 75 97 - 0  
Telefax 0711 / 75 33 16  
[www.euchner.de](http://www.euchner.de) · [info@euchner.de](mailto:info@euchner.de)

**EUCHNER**  
More than safety.