

Electronic-Key-System Manual ActiveX[®] Module Software

Order No. 098 655



EKS.



More than safety.



EUCHNER

Table of contents

1	General notes.....	4
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
2	Support information, installing and uninstalling	5
3	The EKS ActiveX® module	6
3.1	EKS type library	6
3.2	EKS control	6
3.3	Overview of the methods, properties and events in the EKS 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	BaudRate	9
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	Constants.....	15

3.7 Events	16
3.7.1 OnKey	16
3.7.2 OnRead	16
3.7.3 OnWrite	16
4 Examples	17
4.1 Establishing connection with the EKS Electronic-Key adapter.....	17
4.2 Example event call in Visual Basic [®]	18

1 General notes

This ActiveX[®] module supports the integration of the Electronic-Key-System (EKS) Electronic-Key adapter with serial and USB interface into your PC application. EKS can thus be used, e. g., in conjunction with process visualization software. Data communication is in accordance with transfer protocol 3964R. The ActiveX[®] module is used here as a protocol driver.

With the aid of the EKS 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[®] module (order no. 098 708), from version 1.0.1.0.

The manual does not apply to earlier software releases or the previous version of the EKS ActiveX[®] module (order no. 084 708).

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 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, no special requirements

Software: If you use an EKS Electronic-Key adapter with USB interface, the EKS-USB driver (order no. 094 376) from version 1.0.3.0 must be installed on the system.

Operating system: Windows[®] 98
Windows[®] NT
Windows[®] 2000
Windows[®] XP

2 Support information, installing and uninstalling

To be able to use the EUCHNER EKS ActiveX[®] module, you must first install it. For this purpose run the program EKS_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 eks.ocx
- ▶ this manual in Acrobat PDF format

The installation CD contains:

- ▶ the program EKS_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 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 ActiveX[®] module

3.1 EKS type library

- ▶ Description EUCHNER EKS ActiveX module
- ▶ Library EKSLib
- ▶ File name eks.ocx
- ▶ GUID { 62A51CD4-76C1-453D-B258-804D12988851 }
- ▶ Control EKS

3.2 EKS control

- ▶ Control name EKS
- ▶ File name eks.ocx
- ▶ GUID { 64CAE8A8-3CB8-4929-A90F-57499A6E83F3 }
- ▶ Properties 14
- ▶ Events 3
- ▶ Methods 4

Before you can use the EKS ActiveX[®] module in your application, you must add the file eks.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 ActiveX[®] module

The EKS 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	Page
Open	8
Close	8
Read	8
Write	8
Properties	Page
BaudRate	9
Port	9
KeyType	9
LastState	10
StartAdress	11
CountData	11
BlockSize	12
PollingTime	12
Opening	12
Reading	12
Writing	13
KeyState	13
Version	13
Data	13
Debug	14
Events	Page
OnKey	15
OnRead	15
OnWrite	15

3.4 Methods

3.4.1 Open

- ▶ Description Opens the serial interface to the EKS with the properties set (*BaudRate*, *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 get the actual condition of the method *Open*, the property *Opening* can be polled. At the end of a program any open serial connections must be closed again by calling the method *Close*.

3.4.2 Close

- ▶ Description Closes the serial interface 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 serial 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 EKS Electronic-Key, you do not need to explicitly call the method *Read*. As soon as the event *OnKey* is triggered and the property is *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 function 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[®] module, in which the read data of the method *Read* or the event *OnKey* are stored.
- ▶ Syntax: **short = getData** (*short* ByteIndex);
- ▶ Remarks: With the method *getData*, the internal memory area of the ActiveX[®] 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 *StartAdress* 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[®] 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 3.5.14.

3.4.6 setData

- ▶ Description: Writing access to the internal memory area of the ActiveX[®] module, in which the data to be written of the method *Write* are stored.
- ▶ Syntax: **object.setData** (*short* ByteIndex, *short* DataValue);
- ▶ Remarks: With the method *setData*, the internal memory area of the ActiveX[®] module can be written. Upon triggering the event *OnWrite*, the data are written from the temporary storage to the key. In the properties *StartAdress* 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[®] 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 3.5.14.

3.5 Properties

3.5.1 BaudRate

- ▶ Description: Sets the baud rate
- ▶ Syntax: **object.BaudRate = BaudRateConstants** Value;
- ▶ Comments: Using this property the same baud rate must be set as was selected on the EKS using the DIP switches. Possible values are:
EKS_BAUD_9600 = 9600
EKS_BAUD_28800 = 28800
This property is applied by calling the method *Open*.
- ▶ Data type: **BaudRateConstants** (Enumeration)
- ▶ Default value: EKS_BAUD_9600

3.5.2 Port

- ▶ Description Selects the serial interface on the PC
- ▶ Syntax *object.Port = String* Value;
- ▶ Comments Possible values are:
COM1
COM2
...
This property is applied by calling the method *Open*.
- ▶ Data type **String**
- ▶ Default value COM1

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_{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:



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	
0x88	136	Timeout The timeout of approx. 2 s for the 3964R protocol after sending the STX command has been exceeded, the protocol is repeated up to 6 x.
0x89	137	NAKReceived NAK (15 _{hex}) received from EKS -> protocol error
0x8A	138	Collision Collision in the 3964R protocol.
0x8B	139	WrongBaudrate Probably the wrong baud rate has been selected on the EKS or in the ActiveX® module.
0xA0	160	DeviceNotOpened The connection to the EKS has not been opened, please run the method <i>Open</i> .
0xA1	161	DeviceNotAvailable The serial interface selected is not available.
0xA2	162	DeviceInUse The serial interface selected is in use by another application and is not available.
0xB0	176	ReadTimeOut It was not possible to correctly complete the method <i>Read</i> , the method has timed out.
0xB1	177	WriteTimeOut It was not possible to correctly complete the method <i>Write</i> , 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 <i>CountData</i> , is 0.
0xC1	193	NothingToWrite The number of bytes of data to be written, as defined by the property <i>CountData</i> , is 0.
0xD0	208	PortNotOpened The serial interface is not open, please check the setting for the property <i>Port</i> and run the method <i>Open</i> .
0xE0	224	OpenFailed The method <i>Open</i> failed.
0xE1	225	OpenActive The method <i>Open</i> is still active.
0xE6	230	USBReConnected The connection to the EKS USB has been re-established.
0xE7	231	USBDisConnected The connection to the EKS USB has been disconnected.
0xE8	232	Suspend The computer will be placed in the suspend mode.
0xE9	233	ResumeSuspend The suspend mode has been terminated.
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 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 methods 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 during 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 during 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 BlockSize

- ▶ Description Defines the block size for the data transfer.
- ▶ Syntax *object*.BlockSize = short Value;
- ▶ Comments Defines the block size for the data packets in the 3964R protocol. The default value can be left unchanged in this property if no older devices of the EKS series are used.
- ▶ Data type **short**
- ▶ Default value 124

3.5.8 PollingTime

- ▶ Description Defines the time [ms] after which the ActiveX[®] module polls the status of the Electronic-Key from the EKS.
- ▶ Syntax *object*.PollingTime = short Value;
- ▶ Comments The event behavior of the ActiveX[®] module can be changed using this property. If a polling time of 0 ms is set in the property PollingTime, the polling is disabled.
- ▶ Data type **short**
- ▶ Default value 0



Information!

In the normal case polling is not necessary, as the transponder is detected using the CTS signal. Polling is only useful if a CTS signal is not available.

The polling time should not be set too short (< 500 ms), as otherwise there is no time left for other method calls. This situation can result in a frequent *LastState* of *0xFF*.

3.5.9 Opening

- ▶ Description Condition 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

3.5.10 Reading

- ▶ Description Condition 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

3.5.11 Writing

- ▶ Description Condition 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

3.5.12 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

3.5.13 Version

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

3.5.14 Data

- ▶ Description Memory area in which data read by the method *Read* or of 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_{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.

3.5.15 Debug

- ▶ Description If the property *Debug* is set to the value *true*, the COM port is closed at the end of a debug session in a programming environment.
- ▶ Syntax **bool = object.Debug;**
- ▶ Comments The ActiveX[®] 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[®] and Microsoft Excel[®] for the development of the application so that the COM 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, then during the next debug session you will receive a *LastState* of 162 (*DeviceInUse*). If the property *Debug* is *true*, then all COM ports for **all** ActiveX 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[®] module, the COM port will be closed in all other instances.

3.6 Constants

This section lists all constants that are used in the properties for the EKS 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

BaudRateConstants (used in the property *KeyType*)

Value	Constant
9600	EKS_BAUD_9600
28800	EKS_BAUD_28800

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*.

o **Information!**

- II Whether an Electronic-Key has been inserted or removed is detected from the state of the CTS signal (see manual for EKS Electronic-Key adapter).

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 function 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 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 = "COM1"  
EKS.BaudRate = EKS_BAUD_9600  
EKS.KeyType = EKS_KEY_READWRITE  
EKS.PollingTime = 0
```

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

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

3. Open serial 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 Electronic-Key
            ' WARNING! 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[®], Excel[®], ActiveX[®]
and Visual Basic[®] 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 · info@euchner.de

EUCHNER