

SIEMENS

SIMATIC Ident

RFID systems
SIMATIC 3964R protocol
for Windows 9x/XP/7

Installation Manual

Introduction

1

Installation

2

Parameter assignment

3




Programming interface

4

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

 DANGER
indicates that death or severe personal injury will result if proper precautions are not taken.
 WARNING
indicates that death or severe personal injury may result if proper precautions are not taken.
 CAUTION
indicates that minor personal injury can result if proper precautions are not taken.
NOTICE
indicates that property damage can result if proper precautions are not taken.


If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

 WARNING
Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

1	Introduction	5
1.1	Range of functions	6
1.2	System requirements	6
2	Installation	7
2.1	Installing the driver	7
2.2	Adapting the registry	9
2.3	Linking in the header file	9
2.4	Linking in the library file	9
3	Parameter assignment	11
4	Programming interface	13
4.1	Functions of the serial interface	13
4.2	Error and status messages	23
4.3	Linking the driver into your own programs	26

Introduction

Purpose of this document

This manual contains information on the installation and parameter assignment of the "3964R" driver and on programming applications based on the supplied files. With the help of the driver and the supplied header and library files, you can program your own applications to allow communication between Windows PCs and the connected RFID systems.

To install the driver you need to know the Windows version being used Windows XP or Windows 7. To program applications you need to be familiar with the programming languages listed below.

The 3964R driver is implemented as a "DLL" file. To configure the driver functionality there is also an application for the Control Panel in the form of a "CPL" file. You will find the installation files on the DVD "RFID Systems, Software & Documentation" (order number: 6GT2080-2AA20).

- File name of the driver DLL: 3964R.DLL
- File name of the configuration program: CPL3964R.CPL

Programming environment

- Microsoft Visual C++ 6.0, VisualStudio 2010 (C++)
- other C environments with modified header file

1.1 Range of functions

The 3964R driver supports the following functions:

- Support of Windows 9x, XP and Windows 7
- 3964R master/slave protocol or Lauf protocol can be selected
- Any COM port can be set in the parameters (COM1 ... COM255).
Up to 4 serial interfaces can be operated parallel to each other.
- Transmission rate: 300 baud to 256000 baud
- 7 or 8 data bits
- 1, 1.5, 2 stop bits
- Parity freely selectable
- The number of send and send start repetitions can be selected
- Buffer size (send/receive) can be set (in each case up to 65536 bytes)
- Timeout times for 3964R/Lauf can be selected

1.2 System requirements

The software was tested with the following hardware. Operating the driver in an environment with compatible devices is possible without causing problems.

Operating system	System requirements
Windows XP	<ul style="list-style-type: none">• Processor: ≥ 1 GHz• Main memory: ≥ 1 GB RAM
Windows 7	<ul style="list-style-type: none">• Processor: ≥ 2 GHz• Main memory: ≥ 4 GB RAM

Installation

2.1 Installing the driver

You will find the installation files on the DVD "RFID Systems, Software & Documentation" (order number: 6GT2080-2AA20).

Follow the steps below to install the driver:

1. Start the setup program by double-clicking on the ""MOBYAPI.MSI" or "SETUP.EXE" file in the DVD folder "daten/Moby_lib/MOBY_API".

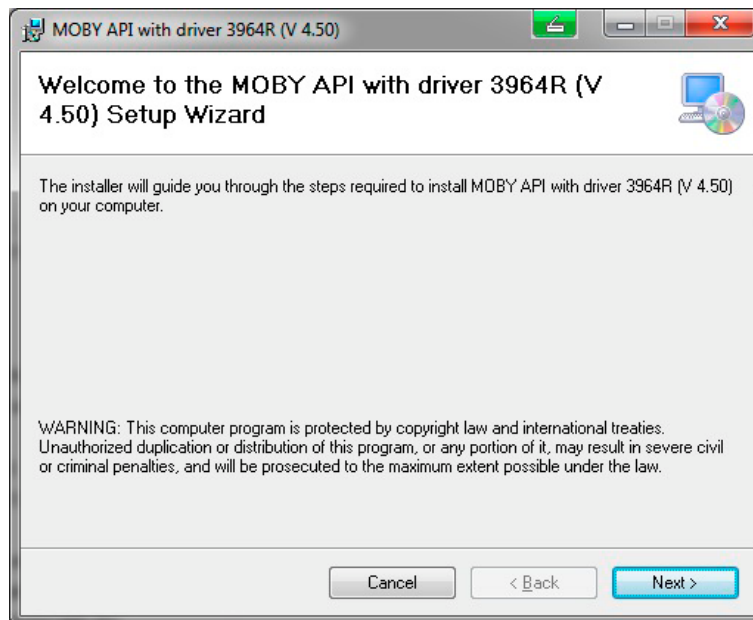


Figure 2-1 Setup program "MOBY API"

2. Follow the instructions of the setup program.

3. Decide whether the driver will be set up for the current user "Just me" or for all users "Everyone".

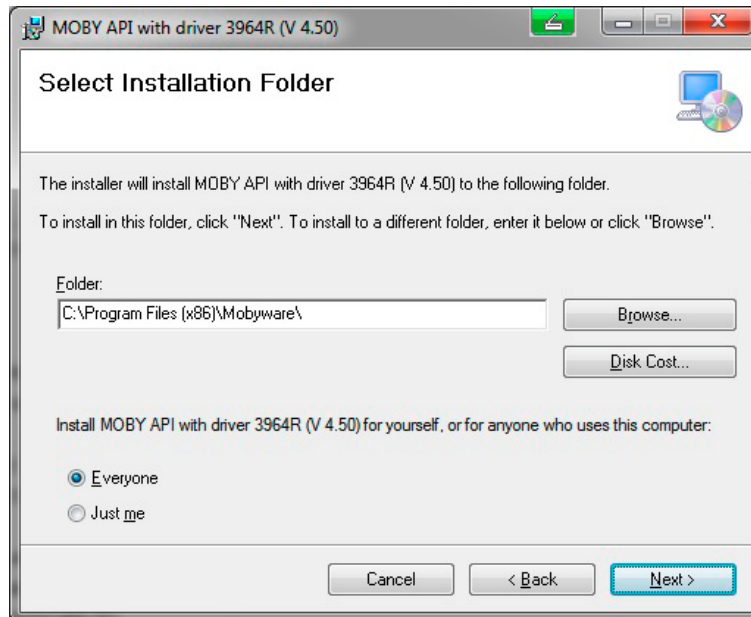


Figure 2-2 Selection of the user group

4. Click the "Next" button to install the driver.
5. Once the driver has been successfully installed, click the "Close" button to exit the setup program.

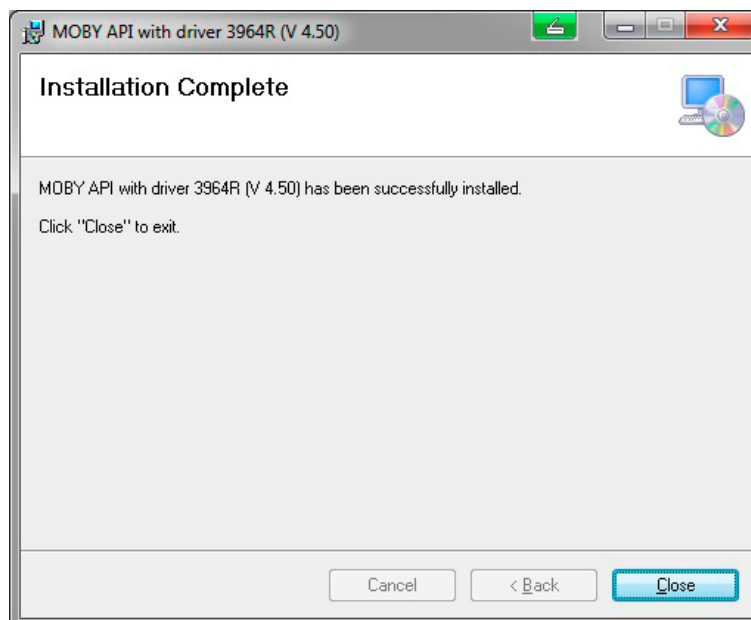


Figure 2-3 Installation successfully completed

After successful installation of the driver the entry "3964R/Lauf" is created in the Control Panel.

2.2 Adapting the registry

When installing the driver, information about the existing interfaces and their configuration is adapted to the Windows Registry and a new registry entry is made. The installation program handles the settings required for this automatically.

2.3 Linking in the header file

If you want to use the function library for program development, you will need to link the header file into the development project.

If you keep to the default installation are when installing the driver, the header file is stored in "C:\Program Files\Mobyware\include".

File name in the header file: 3964R.H

2.4 Linking in the library file

If you want to program your own applications, you will need to link the library file into the corresponding project.

If you keep to the default installation are when installing the driver, the library file is stored in "C:\Program Files\Mobyware\lib".

File name in the library file: 3964R.LIB

Parameter assignment

With the help of the configuration program, you can make parameter settings for the serial interfaces (COM ports). After installation of the driver a new entry "3964R/Lauf" is created in the Control Panel.

Follow the steps below to make settings for the freely selectable interfaces and protocols:

1. Start the configuration program by double-clicking on the "3964R/Lauf" file in the Control Panel.

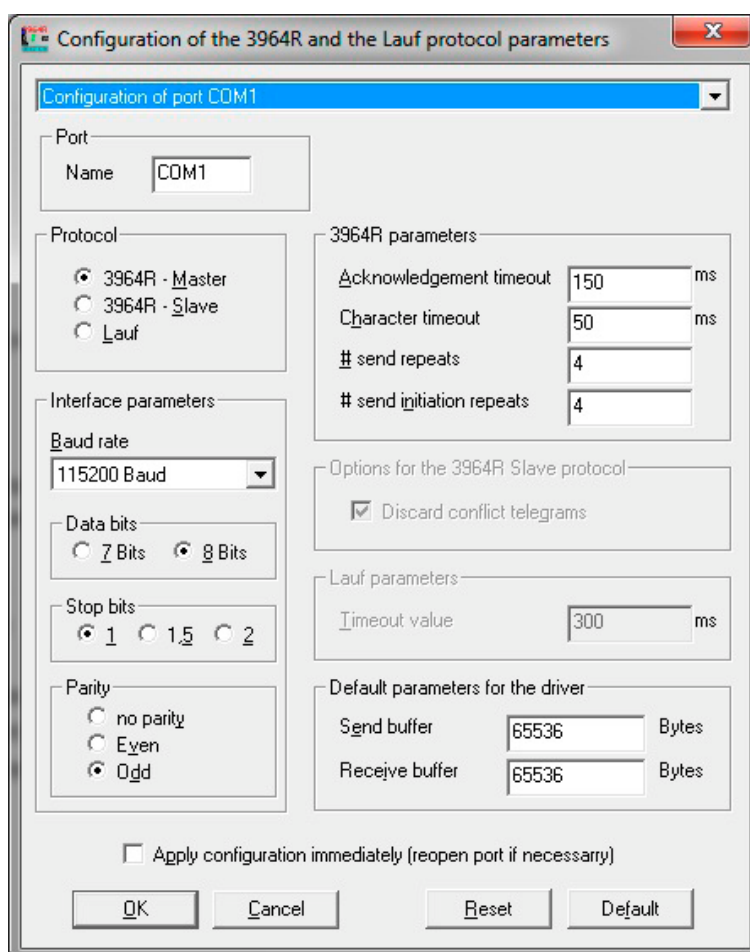


Figure 3-1 Configuration of the COM ports of the 3964R protocol

A maximum of four COM ports can be configured. Each of the four COM ports can be assigned a name to suit your purposes (for example "COM255").

2. From the drop-down list, select the port you want to configure.

3. Enter the name of the interface in the "Name" input box.

The name must match the name specified in the device manager for the connected hardware (integrated serial interface or external adapter, e.g. USB).

4. Select the "Apply configuration immediately" check box to adopt all the data immediately in the configuration.

Close and open the relevant port using the application to adopt the data (e.g. if you change the baud rate). Otherwise, only the data that does not relate to interface parameters will be adopted (e.g. timeout times).

5. Click "OK" to apply your changes.

By clicking the "Reset" button, you reset the parameters to the values that were valid when the configuration program was called. By clicking the "Default" button, you reset the parameters to the original data written to the 3964R as default.

The maximum length for the send and receive buffer is 65536 bytes.

In the "# send repeats" input box, you specify the number of repetitions to be made following a NAK (Negative Acknowledge or parity error). In the "# send initiation repeats" input box, you specify the number of repetitions to be made following an STX (frame start / when the communications partner does not respond).

Programming interface

4.1 Functions of the serial interface

Note

You will find all the error messages listed in this section in the section "Error and status messages (Page 23)".

ComOpen

```
comInt ComOpen (LPCSTR com_name,
                int read_number,
                int write_number,
                HWND hwnd)
```

This function is used to open an interface and to prepare for data transfer. It must be called before all other functions.

Parameter	
com_name	Interface name as string Possible values "COM1 ... COM255"; a maximum of 4 COM ports can be open at the same time
read_number	Size of the receive buffer in bytes
write_number	Size of the send buffer in bytes
hwnd	Windows handle as destination for the notification message

Return value	
≥ 0	Handle for the opened interface
< 0	possible error message: <ul style="list-style-type: none"> • COM_ALREADY_OPEN • COM_DDFINI_ERROR • COM_NO_MEMORY • COM_NO_HANDLE • COM_REGISTRY • COM_NO_CONFIG

4.1 Functions of the serial interface

Notes

- If "COM_OPEN_STD_PUF" is transferred as the buffer size, the buffer size from the parameter assignment dialog is adopted.
- The returned handle must be transferred as a parameter with all further calls.
- Each buffer can hold several frames at the same time. The number is restricted only by the total of the lengths of all buffered frames.

ComRead

```
comInt ComRead (int com_handle,  
                void *read_data,  
                int read_number,  
                long options)
```

This function is used for the blocking receipt of frames.

Parameter	
com_handle	interface handle from ComOpen
read_data	Pointer to the receive buffer of the caller
read_number	Length of the receive buffer
options	Not used

Return value	
≥ 0	Number of bytes read (≠ frame length)
< 0	possible error message: <ul style="list-style-type: none">• COM_HANDLE_FALSE• COM_2SMALL• COM_READ_ERROR

Notes

- The call blocks if no frame is present. No timeout parameter is intended.
- This call fetches incoming frames from an internal buffer. The actual receive routine is asynchronous.

ComWrite

```
comInt ComWrite (int com_handle,
                void *write_data,
                int write_number,
                long options)
```

This function is used to send an asynchronous send job.

Parameter	
com_handle	interface handle from ComOpen
write_data	Pointer to the send buffer of the caller
write_number	Length of the send buffer
options	Not used

Return value	
≥ 0	Number of bytes written (≠ frame length)
< 0	possible error message: <ul style="list-style-type: none"> • COM_HANDLE_FALSE • COM_2SMALL • COM_WRITE_ERROR

Notes

This call places the send job in a send buffer. The actual data communication is asynchronous. As result, the return following the call for this routine simply means that the send job was correctly included in the send queue.

ComEnableEvent

```
comInt ComEnableEvent (int com_handle,
                       int com_event,
                       int user_id,
                       init msg)
```

This function enables events (for checking the asynchronous transfer).

4.1 Functions of the serial interface

Parameter	
com_handle	interface handle from ComOpen
com_event	Event to be enabled possible functions: COM_READ_EVENT, COM_WRITE_EVENT (both ORed)
user_id	ID to be assigned by the user
msg	Message number to be assigned by the user

Return value	
X	Screen with the current events prior to the call

Notes

- If an asynchronous send or receive call is completed and if the corresponding event is activated using this routine, a Windows message is sent to the window with the handle from ComOpen with the message ID "msg". The data of this message can be queried with the help of "COMGetNotify". The "user_id" is sent with every message for simpler message identification.
- "user_id" and "msg" are used only for the events specified in this call. By calling the "COMEnableEvent" function several times it is therefore possible to combine various message types for reading and writing.

ComDisableEvent

```
comInt ComDisableEvent (int com_handle,
                        int com_event)
```

This function disables events (for checking the asynchronous transfer).

Parameter	
com_handle	interface handle from ComOpen
com_event	Event to be disabled possible functions: COM_READ_EVENT, COM_WRITE_EVENT (both ORed)

Return value	
X	Screen with the current events prior to the call

ComGetNotify

```
comInt ComGetNotify (WPARAM wParam,
                    LPARAM lParam,
                    int *user_id,
                    int *event_ptr,
                    int *state_ptr,
                    int *handle_ptr)
```

Messages that are sent to the application at the end of communications operations (if they have been enabled with "COMEnableEvent") contain further information about the triggering operation. These are coded in the message parameters "wParam" and "lParam" and can be extracted using this function.

Parameter	
wParam	"wParam" parameter of the incoming message
lParam	"lParam" parameter of the incoming message
*user_id	Pointer to storage space for the ID assigned by the user when calling "COMEnableEvent"
*event_ptr	Pointer to the storage location for event type
*state_ptr	Pointer to the storage location for the status of the completed operation that triggered the message
*handle_ptr	Pointer to storage location for the interface handle from ComOpen

Return value	
0	Function successfully completed
≠ 0	possible error message: <ul style="list-style-type: none"> • COM_UNKNOWN_EVENT

Notes

- If "NULL" is transferred with any pointer parameter, the corresponding information is not returned.
- The routine runs several basic plausibility checks of the values of the message parameters, a full check is however not possible.

ComSetNotification

```
comInt ComSetNotification (comHandle_t com_handle,
                          comNotifCall p_callback,
                          int userID)
```

With this function, a callback routine can be specified that is called if events occur in the driver (e.g. received frame). If the "NULL" is transferred as the parameter, the routine is disabled. This notification method can be considered as an alternative to "ComEnableEvent".

Parameter	
com_handle	interface handle from ComOpen
com_NotifCall	Pointer to the callback routine
userID	User-defined ID (any value)

Return value	
0	Function successfully completed
< 0	possible error message: <ul style="list-style-type: none"> • COM_HANDLE_FALSE

ComGetReadState

```
comInt ComGetReadState (int com_handle)
```

This function queries the current status of the receive routine and then resets it.

Parameter	
com_handle	interface handle from ComOpen

Return value	
COM_ST_FREE	Receive routine is not busy
COM_ST_BUSY	A frame is currently being received
COM_ST_SUCCESS	A frame was successfully received
< 0	Error status

Notes

- If the current status is "COM_ST_BUSY", the status is not reset.
- The error status can include several errors at the same time. Each bit in the error number corresponds to an error. The "COM_ST_ERROR" is set in any case. Section "Error and status messages (Page 23)" contains a precise description of the individual errors.

ComGetWriteState

comInt ComGetWriteState (int com_handle)

This function is used to query the current status of the send routine and then to reset it.

Parameter	
com_handle	interface handle from ComOpen

Return value	
COM_ST_FREE	Send routine is not busy
COM_ST_BUSY	A frame is currently being sent
COM_ST_SUCCESS	A frame was sent successfully
< 0	Error status

Notes

- If the current status is "COM_ST_BUSY", the status is not reset.
- The error status can include several errors at the same time. Each bit in the error number corresponds to an error. The "COM_ST_ERROR" is set in any case. Section "Error and status messages (Page 23)" contains a precise description of the individual errors.

ComClose

comInt ComClose (int com_handle)

This function closes an interface and releases all the resources associated with it.

Parameter	
com_handle	interface handle from ComOpen

Return value	
0	Function successfully completed
≠ 0	possible error message: <ul style="list-style-type: none">• COM_NOT_OPEN

Notes

Following this call, no data transfer routines of this driver can be used until the next the "ComOpen".

ComGetVersion

comInt ComGetVersion (char *ver_string)

This function returns the version number of the DLL being used and an expanded version string (if required).

Parameter	
ver_string	Buffer for the expanded version string No expanded version information if "NULL".

Return value	
X	Version number (version x.y → x × 100 + y)

Notes

- The current version returns the value "450" for version 4.50 and the following expanded version string "3964R / Lauf for Windows 95/NT/XP/7, version 4.50, November 2012".
- The transferred buffer area should have a minimum size of 100 characters.

ComString

comInt ComString (char *errs,
int error,
int typ)

This function returns an error text for the transferred error value.

Parameter	
errs	Buffer for error text
error	Error number (result of a function of the user programming interface)
typ	possible error message: <ul style="list-style-type: none"> • COM_STR_OPEN = ComOpen • COM_STR_RDWR = ComRead or ComWrite • COM_STR_STATE = ComReadState or ComWriteState • COM_STR_EVENT = ComEnableEvent or ComDisableEvent • COM_STR_CLOSE = ComClose

Return value	
X	0

Notes

The transferred buffer should have a minimum size of 500 characters.

ComReadConfig

`comInt ComReadConfig` (LPCSTR devName,
devConfig_p conf)

This function reads the configuration data for a specified port from the registry and returns it in an edited form. The port does not need to be opened.

Parameter	
devName	Interface name as string possible values: "COM1 ... COM255"
conf	Pointer to configuration variable

Return value	
0	Function successfully completed
< 0	possible error message: <ul style="list-style-type: none"> • COM_REGISTRY • COM_NO_CONFIG

ComWriteConfig

comInt ComWriteConfig (LPCSTR devName,
devConfig_p conf,
BOOL force)

This function writes the configuration data for a specified port to the registry. The port does not need to be opened.

Parameter	
devName	Interface name as string possible values: "COM1 ... COM255"
conf	Pointer to configuration variable
force	Force the immediate adoption of the configuration

Return value	
0	Function successfully completed
< 0	possible error message: <ul style="list-style-type: none">• COM_REGISTRY• COM_NO_CONFIG

Notes

If the "force" parameter is set to "TRUE", the configuration data is adopted immediately even if the interface is already open. If necessary, the interface is closed to allow this and then opened again.

4.2 Error and status messages

Messages returned by the functions described in the previous section can be divided into three classes:

- Positive status messages
- Status error messages
- System error messages

The following error codes are defined in the header file "3964r.h".

Positive status messages

- Used by:
- ComReadState
 - ComWriteState

The status messages are returned when:

- an operation was completed successfully
- an event was not evaluated as an error

Status messages	
COM_ST_FREE	Everything OK, no job present
COM_ST_BUSY	A job is currently being processed, no error has occurred up to now.
COM_ST_SUCCESS	The last operation was completed successfully.

Status error messages

- Used by:
- ComReadState
 - ComWriteState

These messages signal status errors that occurred during the transfer of frames. They mostly relate to the interface or local system resources.

Each message can include several individual messages. This is achieved by ORing the individual error messages. To identify a status error message, however, the "COM_ST_ERROR" is always set. The following status error messages are possible:

Status error messages	
COM_ST_2MANY	Too many jobs, send queue full
COM_ST_NO_CON	Connection establishment failed (sending)
COM_ST_NO_TRA	Frame transfer failed (sending)
COM_ST_2SMALL	Receive buffer too small when receiving
COM_ST_BCCERR	Block check error while receiving
COM_ST_TIMCON	Timeout during connection establishment (sending)
COM_ST_TIMTRA	Timeout during a frame (receiving)
COM_ST_TIMQUI	Timeout during connection termination (sending)
COM_ST_SCC_BR	COM message: Break
COM_ST_SCC_PY	COM message: Parity error
COM_ST_SCC_FR	COM message: Framing error
COM_ST_SCC_OR	COM message: Overrun
COM_ST_SNDRCV	Duplex conflict (sending)
COM_ST_SYSERR	Unknown system error

System error messages

- Used by:
- ComOpen
 - ComRead
 - ComWrite
 - ComGetNotify
 - ComClose

Messages of this class show that function calls have failed and usually mean a serious error in the system and/or in the configuration settings.

This protocol implementation uses only a subset of the error messages listed in the header file. All the relevant error messages are listed below:

System error messages	
COM_HANDLE_FALSE	Transferred handle is invalid
COM_ALREADE_OPEN	Interface is already open
COM_DDFINI_ERROR	Interface initialization error
COM_NO_MEMORY	Not enough memory available
COM_2SMALL	Buffer overflow
COM_READ_ERROR	Bad receive job
COM_WRITE_ERROR	Bad send job
COM_UNKNOWN_EVENT	Message parameter invalid
COM_NOT_OPEN	Interface is not open
COM_NO_HANDLE	No further port can be opened
COM_REGISTRY	Error accessing the Windows registry
COM_NO_CONFIG	The addressed port is not configured

4.3 Linking the driver into your own programs

To use this programming interface for the 3964R protocol in your own programs, note the following two steps:

- The "3964R.H" header file must be linked into your own source code using the "#include" preprocessor command. This declares all function calls and constants.
- The actual library "3964R.LIB" must be also linked in when linking.

The header file was developed with and for Visual C++ from Microsoft. If you want to use a different programming language or a C++ dialect from another company, you will need to adapt the header file accordingly.