SIEMENS

Table of Contents

MOBY

3964R Protocol under Windows NT 4.0/95

User's Manual

J31069-D0131-U001-A0-7618

Release 06.99

Safety Guidelines

This manual contains notices which you should observe to ensure your own personal safety, as well as to protect the product and connected equipment. These notices are highlighted in the manual by a warning triangle and are marked as follows according to the level of danger:

⚠ **Danger**

indicates that death, severe personal injury or substantial property damage will result if proper precautions are not taken.

⚠ **Warning**

indicates that death, severe personal injury or substantial property damage can result if proper precautions are not taken.

⚠ **Caution**

indicates that minor personal injury or property damage can result if proper precautions are not taken.

Note

draws your attention to particularly important information on the product, handling the product, or to a particular part of the documentation.

Qualified Personnel

The device/system may only be set up and operated in conjunction with this manual. Only **qualified personne**l should be allowed to install and work on this equipment. Qualified persons are defined as persons who are authorized to commission, to ground, and to tag circuits, equipment, and systems in accordance with established safety practices and standards.

Trademarks

MOBY® is registered trademark of SIEMENS AG.

Some of the other designations used in these documents are also registered trademarks; the owner's rights may be violated if they are used be third parties for their own purposes.

Siemens AG
Automation & Drives
Systems Engineering
Postfach 2355, D-90713 Fuerth

Technical data subject to change.

Siemens Aktiengesellschaft

Order No. J31069-D0131-U001-A0-7618

Table of Contents

1 Introduction

1.1 Scope of Functions

- Support of Windows NT 4.0 and Windows 95
- 3964R master/slave protocol or Lauf protocol can be selected.
- COM 1 to COM 4 can be parameterized.
  Up to 4 serial interfaces can be operated in parallel.
- Transmission speed: 300 Baud to 38400 Baud
- 7 or 8 data bits
- 1, 1.5, 2 stop bits
- Parity can be selected as desired.
- Number of sending and sending-start repetitions can be selected.
- Timeout times for 3964R/Lauf can be selected.

1.2 System Prerequisites

The software was tested with the following hardware. Use of the driver in an environment with compatible devices presents no problems.

- IBM-compatible PC with 133 MHz Pentium CPU
- 32-MB main memory
- Windows NT 4.0

or

- IBM-compatible PC with x486 DX2-66 CPU
- 8-MB main memory
- Windows 95

Programming environment

- Microsoft Visual C ++
- Other C environments with modified header file
- Other programming languages (e.g., Visual Basic) using a wrapper

2

Installation

2.1 Installing the Driver

The 3964R protocol is provided as a DLL file. In addition, there is an application in the form of a CPL file for system control for configuration of driver functionality.

File names:	3964R.DLL	Driver DLL
	CPL3964R.CPL	Configuration program

For installation, the CPL file and the DLL file must be copied to the following directory.

Under **Windows NT 4.0:**	C:\WINNT\SYSTEM32
Under **Windows 95:**	C:\WIN95\SYSTEM

C:\WINNT or C:\WIN95 must be replaced by the applicable directory of the installation used.

The actual driver DLL must be copied to a directory which is entered in the environment variable PATH. For example, this can also be the OS system path in which the CPL file is already located.

2.2 Adapting the Registry

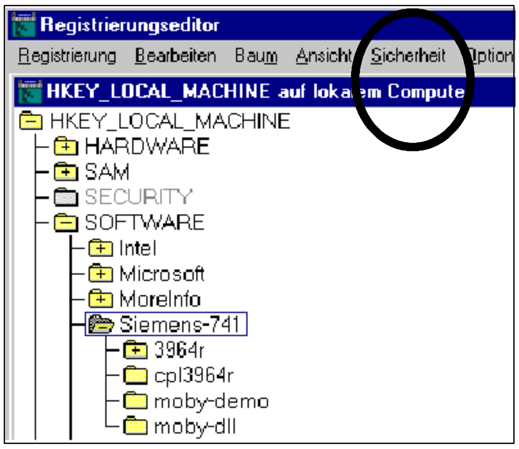
The driver DLL receives its information on existing interfaces and their configuration via the Windows registry. To prepare this for use of the 3964R driver, the entries specified in the SIEMENS.REG file must be entered in the registry. This can be done directly by calling this file (i.e., double click the SIEMENS.REG file).

Special features under WINDOWS NT

How does a user under Windows NT obtain the rights to the 3964R driver when the driver has been installed by the administrator?

1. Start the program REGEDT32.EXE.

2. In the window HKEY_LOCAL_MACHINE (see figure 2-1), open the path “Siemens-741.”

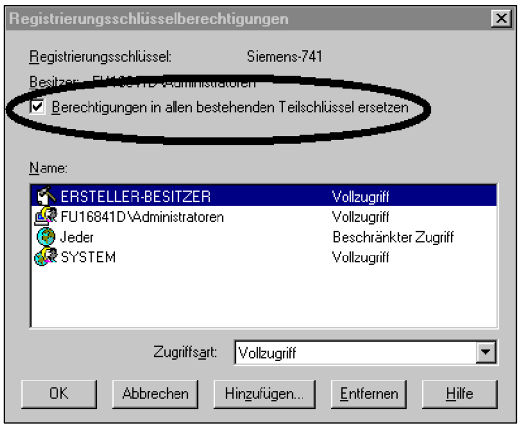


HKEY_LOCAL_MACHINE auf lokalem Compute

```
HKEY_LOCAL_MACHINE
├ HARDWARE
├ SAM
├ SECURITY
├ SOFTWARE
│  ├ Intel
│  ├ Microsoft
│  ├ MoreInfo
│  ├ Siemens-741
│  │  ├ 3964r
│  │  ├ cpl3964r
│  │  ├ moby-demo
│  │  └ moby-dll
```

Figure 2-1 HKEY_LOCAL_MACHINE window

3. Select “Sicherheit” in the menu bar. See figure 2-1. The window “Registrierungsschlüsselberechtigungen” appears. In this window, enable the option “Berechtigungen...”. See figure 2-2.



Registrierungsschlüssel: Siemens-741

Besitzer: FU16841D\Administratoren

☑ Berechtigungen in allen bestehenden Teilschlüssel ersetzen

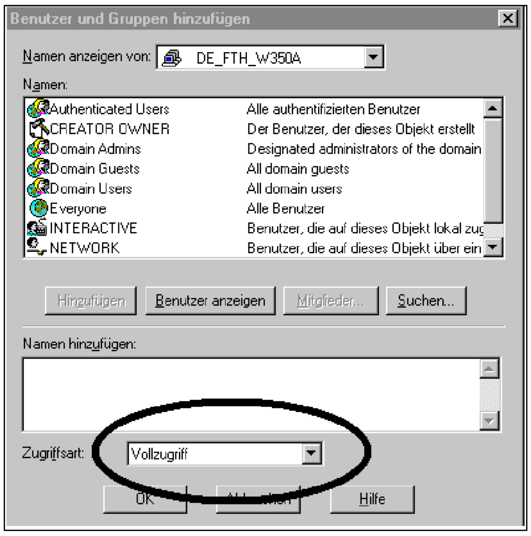
Name:

ERSTELLER-BESITZER	Vollzugriff
FU16841D\Administratoren	Vollzugriff
Jeder	Beschränkter Zugriff
SYSTEM	Vollzugriff

Zugriffsart: Vollzugriff

Figure 2-2 ”Registrierungsschlüsselberechtigungen” window

4. Select the button “Hinzufügen.” In the window which appears, change “Zugriffsart” to “Vollzugriff.” See figure 2-3. Then close all windows with OK.



Namen anzeigen von: DE_FTH_W350A

Namen:

Authenticated Users	Alle authentifizierten Benutzer
CREATOR OWNER	Der Benutzer, der dieses Objekt erstellt
Domain Admins	Designated administrators of the domain
Domain Guests	All domain guests
Domain Users	All domain users
Everyone	Alle Benutzer
INTERACTIVE	Benutzer, die auf dieses Objekt lokal zug
NETWORK	Benutzer, die auf dieses Objekt über ein

Namen hinzufügen:

Zugriffsart: Vollzugriff

Figure 2-3 ”Benutzer und Gruppen hinzufügen” window

2.3 Installing the Header File

If the library is to be used in addition to program development, the header file must also be copied to the development project.

File name: 3964R.H

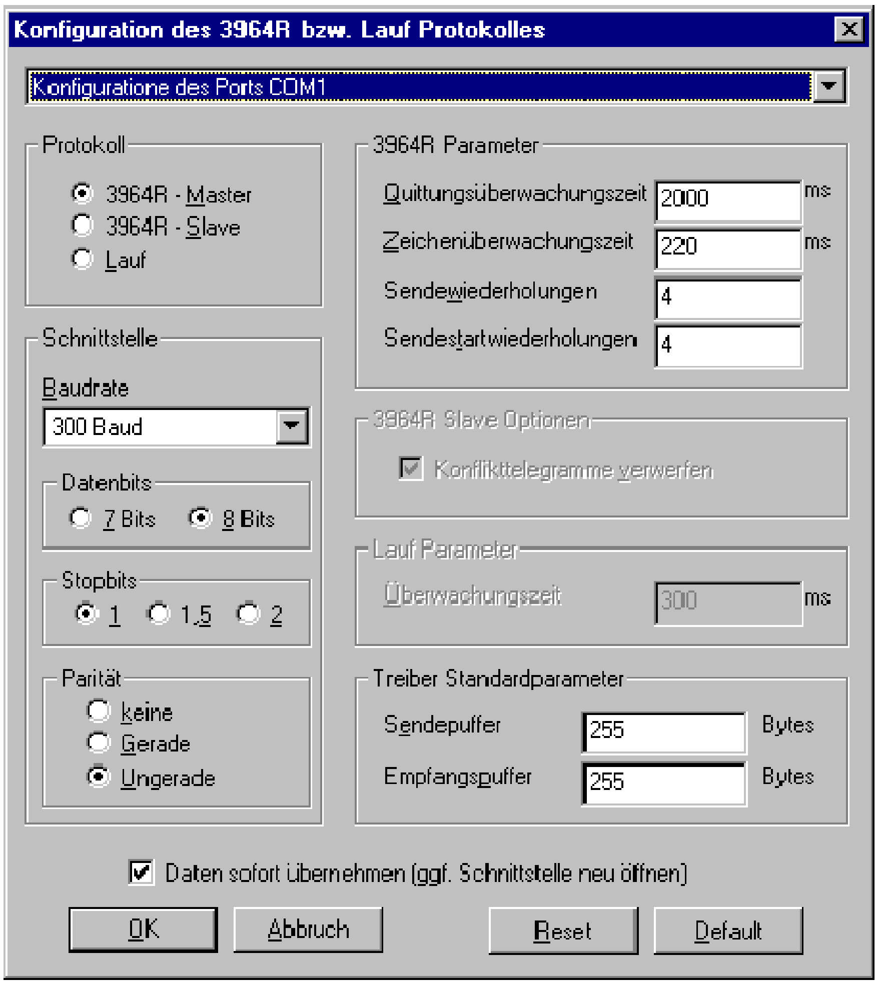
2.4 Installing the Library

The library file must still be installed in addition to program development. This file can be copied to any location where it must be linked from there to the applicable project.

File name: 3964R.LIB

Parameterization

The following dialog window is used to parameterize the freely selectable interface and protocol settings. This can be started directly from system control.



Konfiguration des 3964R bzw. Lauf Protokolles

Konfiguratione des Ports COM1

Protokoll
- [✓ 3964R - Master]
- 3964R - Slave
- Lauf

3964R Parameter

Quittungsüberwachungszeit	2000	ms
Zeichenüberwachungszeit	220	ms
Sendewiederholungen	4	
Sendestartwiederholungen	4	

Schnittstelle

Baudrate: 300 Baud

Datenbits: 7 Bits [✓ 8 Bits]

Stopbits: [✓ 1] 1,5 2

Parität
- keine
- Gerade
- [✓ Ungerade]

3964R Slave Optionen

[✓ Konfliktelegramme verwerfen]

Lauf Parameter

Überwachungszeit 300 ms

Treiber Standardparameter

Sendepuffer	255	Bytes
Empfangspuffer	255	Bytes

[✓ Daten sofort übernehmen (ggf. Schnittstelle neu öffnen)]

Figure 3-1 Dialog window for configuration of the protocol

The drop-down menu at the top of the dialog window can be used to select the interface to be configured.

If the checkbox “Daten sofort übernehmen” is activated, all changed data are accepted immediately in the configuration even when the port must be closed and opened (e.g., when the baud rate is changed). Otherwise, only the data are accepted which do not affect any interface parameters (e.g., timeout times).

The “Default” button resets the parameters to their original settings specified in the 3964R standard.

The “Reset” button resets the parameters to the values which were valid when the configuration program was called.

The maximum length for the sending and receiving buffer is 255 bytes.

“Sendewiederholungen” is the number of repetitions due to an NAK (i.e., Negative Acknowledge or parity error).

“Sendestartwiederholungen” is the number of repetitions of an STX (i.e., telegram start) when the communication partner does not respond.

Note

The MOBY user data length may be set to a maximum size of 248 bytes for the data transmission.

4 Programming Interface

4.1 Functions of the Serial Interface

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 it for the data transfer. It must be called before all other functions.

Parameters:

com_name	Interface name as string Possible values: “COM1” to “COM4”
read_number	Size of the receiving buffer in bytes
write_number	Size of the sending buffer in bytes
hwnd	Windows handle as destination of the confirmation messages

Return values:

>=0	Handle for opened interface	
< 0	Error message. See chap. 4.2.	
	Possible:	COM_ALREADY_OPEN COM_DDFINI_ERROR COM_NO_MEMORY COM_NO_HANDLE COM_REGISTRY COM_NO_CONFIG

Remarks:

- When COM_OPEN_STD_PUF is transferred as the buffer size, the applicable buffer size is taken from the parameterization dialog.
- The returned handle must be transferred as a parameter to all other calls.
- Each buffer can hold several telegrams at the same time. The number is only limited by the sum of the lengths of all stored telegrams.

ComRead

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

This function is used to block receipt of telegrams.

Parameters:

com_handle	Interface handle from ComOpen
read_data	Pointer to the receiving buffer of the caller
read_number	Length of the receiving buffer
options	Not used

Return values:

>=0	Number of bytes read = telegram length
< 0	Error message. See chap. 4.2. Possible: COM_HANDLE_FALSE COM_2SMALL COM_READ_ERROR

Remarks:

- The call is blocked if no telegram is present. A timeout parameter is not provided.
- This call fetches incoming telegrams from an internal buffer. The actual receiving routine is asynchronous.

ComWrite

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

This function is used to issue an asynchronous sending job.

Parameters:

com_handle	Interface handle from ComOpen
write_data	Pointer to the sending buffer of the caller
write_number	Length of the sending buffer
options	Not used

Return values:

>= 0	Number of bytes written = telegram length
< 0	Error message. See chap. 4.2. Possible: COM_HANDLE_FALSE COM_2SMALL COM_WRITE_ERROR

Remarks:

- This call places the sending job in a sending buffer. The actual data communication is asynchronous. Return after this routine has been called only means that the sending job was correctly added to the sending waiting queue.

ComEnableEvent

comInt *ComEnableEvent* (int com_handle, int com_event, int user_id, int msg)

This function is used to enable events (to check the asynchronous transmission).

Parameters:

com_handle	Interface handle from ComOpen
com_event	Event to be enabled Possible: COM_READ_EVENT COM_WRITE_EVENT (Both OR-linked)
user_id	ID to be assigned by the user
msg	Message number to be assigned by the user

Return values:

Always	Screen of the current events before the call

Remarks:

- In case an asynchronous sending or receiving call is concluded and the applicable event is enabled with this routine, a Windows NT message is sent to the window with the handle from “ComOpen” with the message ID “msg.” The data of this message can be scanned with “ComGetNotify.” “user_id” is sent to make identification of each message easier.
- Since “user_id” and “msg” are only used for the events specified in this call, different message types can be specified for reading and writing by calling the “ComEnableEvent” function several times.

ComDisableEvent

comInt *DisableEvent* (int com_handle, int com_event)

This function is used to disable events (to check the asynchronous transmission).

Parameters:

com_handle	Interface handle from ComOpen
com_event	Event to be disabled Possible: COM_READ_EVENT COM_WRITE_EVENT (Both OR-linked)

Return values:

Always	Screen of the current events before the call

ComGetNotify

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

Messages, which are sent to the application after communication operations are concluded (if enabled with “ComEnableEvent”), contain additional information on the status of the triggering operation. These are coded in the message parameters “wParam” and “lParam” and can be extracted with this function.

Parameters:

wParam	“wParam” parameter of the incoming message
lParam	“lParam” parameter of the incoming message
user_id	Pointer to the memory location for the ID specified by the user in the “ComEnableEvent” call
event_ptr	Pointer to the memory location for event type
state_ptr	Point to the memory location for the status of the concluded operation which triggered the message
handle_ptr	Pointer to the memory location for the interface handle from ComOpen

Return values:

0	Everything OK
Other	Error message. See chap. 4.2. Possible: COM_UNKNOWN_EVENT

Remarks:

- If NULL is transferred with any pointer parameter, the applicable information is not returned.
- Although the routine performs several basic plausibility tests on the values of the message parameters, a complete check is not possible.

ComSetNotification

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

This function is used to specify a callback routine which is called when events occur on the driver (e.g., telegram received). If NULL is transferred as the parameter, the routine is disabled. This method of notification can be considered as an alternative to “ComEnableEvent.”

Parameters:

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

Return values:

0	Everything OK	
< 0	Error message. See chap. 4.2.	
	Possible:	COM_HANDLE_FALSE

ComGetReadState

comInt *ComGetReadState* (int com_handle)

This function is used to scan the current status of the receiving routine and then reset it.

Parameters:

com_handle	Interface handle from ComOpen

Return values:

COM_ST_FREE	Receiving routine is not busy.
COM_ST_BUSY	A telegram is being received.
COM_ST_SUCCESS	A telegram was successfully received.
< 0	Error status. See chap. 4.2.

Remarks:

- If the current status is COM_ST_BUSY, the status is not reset.
- The error status can contain several errors simultaneously. Each bit in the error number corresponds to an error. However, the COM_ST_ERROR bit is always set. Chapter 4.2 contains a detailed description of the individual errors.

ComGetWriteState

comInt *ComGetWriteState* (int com_handle)

This function is used to scan the current status of the sending routine and then reset it.

Parameters:

com_handle	Interface handle from ComOpen

Return values:

COM_ST_FREE	Sending routine is not busy.
COM_ST_BUSY	A telegram is being sent.
COM_ST_SUCCESS	A telegram was successfully sent.
< 0	Error status. See chap. 4.2.

Remarks:

- If the current status is COM_ST_BUSY, the status is not reset.
- The error status can contain several errors simultaneously. Each bit in the error number corresponds to an error. However, the COM_ST_ERROR bit is always set. Chapter 4.2 contains a detailed description of the individual errors.

ComClose

comInt *ComClose* (int com_handle)

This function is used to close an interface and release all resources connected to it.

Parameters:

com_handle	Interface handle from ComOpen

Return values:

0	Successful
Other	Error message. See chap. 4.2. Possible: COM_NOT_OPEN

Remarks:

- After this call, no data transmission routines of this driver can be used until the next “ComOpen.”

ComGetVersion

comInt *ComGetVersion* (char *ver_string)

This function supplies the version number of the DLL used plus an expanded version string (if desired).

Parameters:

ver_string	Buffer accepting the expanded version string No expanded version information if NULL

Return values:

Always	Version number (version x.y → x*100 + y)

Remarks:

- The present version returns 300 for version 3.00 and “3964R/Lauf auf Windows NT, Version 3.00, Juli 1998” for the expanded version string.
- The transferred buffer area must have at least 100 characters.

ComString

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

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

Parameters:

errs	Buffer for error text			
error	Error number (result of a function of the user programming interface)			
typ	Function which caused the error			
	Possible:	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 values:

Always	0

Remarks:

- The transferred buffer must have at least 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 as conditioned. The port does not have to be open.

Parameters:

devName	Interface name as string Possible values: “COM1” to “COM4”
conf	Pointer to configuration variable

Return values:

0	Successful
< 0	Error message. See chap. 4.2. Possible: COM_REGISTRY COM_NO_CONFIG

ComWriteConfig

comInt *ComWriteVersion* (LPCSTR devName, devConfig_p conf, BOOL force)

This function writes the configuration data for a specified port in the registry. The port does not have to be open.

Parameters:

devName	Interface name as string Possible values: “COM1” to “COM4”
conf	Pointer to configuration variable
force	Forces immediate acceptance of the configuration

Return values:

0	Successful
< 0	Error message. See chap. 4.2. Possible: COM_REGISTRY COM_NO_CONFIG

Remarks:

- If the “force” parameter is set to TRUE, the configuration data are accepted immediately even when the interface is already open. If necessary, the interface is also closed and opened again.

4.2 Error and Status Messages

Messages supplied by the functions described above can be divided into three classes (i.e., positive status messages, error status messages and system error messages).

Positive status messages

Used by:	ComReadState ComWriteState

These messages are returned under the following conditions.

- When an operation is successful
- When an event occurs which is not considered an error

COM_ST_FREE	Everything OK. No job present.
COM_ST_BUSY	A job is being processed. No errors up to now.
COM_ST_SUCCESS	The last operation was concluded successfully.

Error status messages

Used by:	ComReadState ComWriteState

These messages signal status errors which occur during transmission of telegrams. They are usually related to the interface or the local system resources.

Each message can contain several individual messages. This is done by OR-linking the individual error messages. The COM_ST_ERROR bit is always set to indicate an error status message. Other possible individual messages are listed below.

COM_ST_2MANY	Too many jobs. Sending waiting queue is full.
COM_ST_NO_CON	Connection establishment unsuccessful (sending)
COM_ST_NO_TRA	Telegram transfer unsuccessful (sending)
COM_ST_2SMALL	Receiving buffer too small for receiving
COM_ST_BCCERR	Block check error during receiving
COM_ST_TIMCON	Timeout for connection establishment (sending)
COM_ST_TIMTRA	Timeout during a telegram (receiving)
COM_ST_TIMQUI	Timeout during connection establishment (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 indicate the malfunction of function calls and usually mean a serious error in the system and/or in the configuration settings.
This protocol implementation uses only a portion of the error messages shown in the header file. Only the relevant messages are listed below.

COM_HANDLE_FALSE	Transferred handle is invalid.
COM_ALREADY_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	Erroneous receiving job
COM_WRITE_ERROR	Erroneous sending job
COM_UNKNOWN_EVENT	Message parameter invalid
CON_NOT_OPEN	Interface is not open.
COM_NO_HANDLE	No further ports can be opened.
COM_REGISTRY	Error while accessing the Windows registry
COM_NO_CONFIG	The addressed port is not configured.

4.3 Linking to Own Programs

The following two steps are required to use this programming interface for the 3964R protocol with your own programs.

- The 3964R.H header file must be linked in your own source code via the preprocessor command “#include.” This declares all function calls and constants.
- The actual 3964R.LIB library must be linked during linking.

The header file was developed with and for Visual C++ from Microsoft. If another programming language or a C++ version of another company is to be used, the header file must be adjusted.

MOBY 3964R Protocol
J31069-D0131-U001-A0-7618