Siemens S5 (3964R) Driver Help

Table of Contents

Siemens S5 (3964R) Driver Help

Help version 1.017

CONTENTS

Overview

The Siemens S5 (3964R) Driver provides an easy and reliable way to connect Siemens S5 (3964R) devices to OPC Client applications, including HMI, SCADA, Historian, MES, ERP and countless custom applications. It is intended for use with Siemens S5 PLCs communicating via a communications processor card (such as the CP 544) configured to use the 3964R. It can also be used with 3964 protocols and the RK 512 computer link. Multiple CPU systems are supported.

Note: This driver is not designed to respond to unsolicited data from the PLC.

Device Setup

Supported Devices
Any device that supports 3964 or 3964R protocol and uses the RK 512 computer link program.

Communication Protocols
3964R
3964

Note: The 3964 variant is identical to 3964R, except it does not use a Byte Check Character.

Supported Communication Parameters
Values depend on the communications processor card used and its configuration. The following values are typical settings:

Baud: 300 to 19200
Parity: Even
Data Bits: 8
Stop Bits: 1

Device Configuration
The device must be configured to operate in slave mode with low priority partner.

Unsolicited Messages
This driver will accept and acknowledge unsolicited messages from the PLC, but does not use them. For optimum driver performance, unsolicited messages are discouraged.

Cable Connections
Communication processor cards that are compatible with the S5 PLC offer many connection options. A typical setup using a CP 525 card and the V.24 sub module is illustrated below. For more information, refer to the communication processor card's documentation.
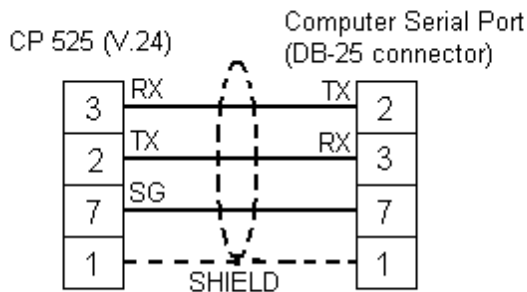
NOTE:
Do not connect the signal ground and shield together.
This could damage the submodule.

Modem Setup

This driver supports modem functionality. For more information, please refer to the topic "Modem Support" in the OPC Server Help documentation.

Protocol Options

The Protocol Options dialog is used to specify the protocol variation, CPU number, handshaking flag parameters and byte order.

Descriptions of the parameters are as follows:

- **Use BCC (3964R):** Most device configurations utilize the 3964R protocol version, which uses a Byte Check Character as a means of communication error checking. In rare cases, the 3964 protocol version may be utilized. This version is identical to 3964R but does not use the Byte Check Character.
- **Specify CPU number in packet:** If the system has multiple CPUs, the message packets must specify the CPU with which to communicate. To do so, check this box and then enter the CPU number in the corresponding field. If only one CPU is used, this box may be left unchecked.
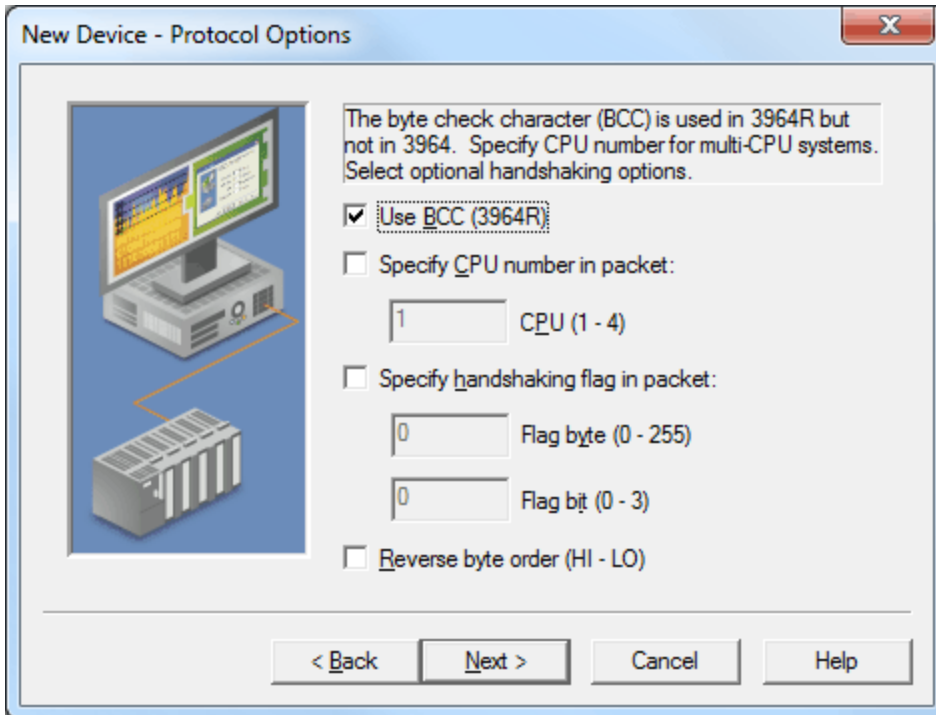- **Specify handshaking flag in packet:** If the configuration requires the message packets to specify handshaking flag parameters, check this box. Then, enter the Flag Byte and Flag Bit values in the corresponding fields.
- **Reverse byte order (HI-LO):** Siemens devices normally store multi-byte data values from least to most significant byte (LO to HI). If the device is programmed to store data in the opposite order, check this box.

    Note: This setting will not affect string data.

String Options

The String Options dialog is used to specify how string data is passed to client applications. The driver reads all of the bytes in the memory range specified by a String Tag's address.
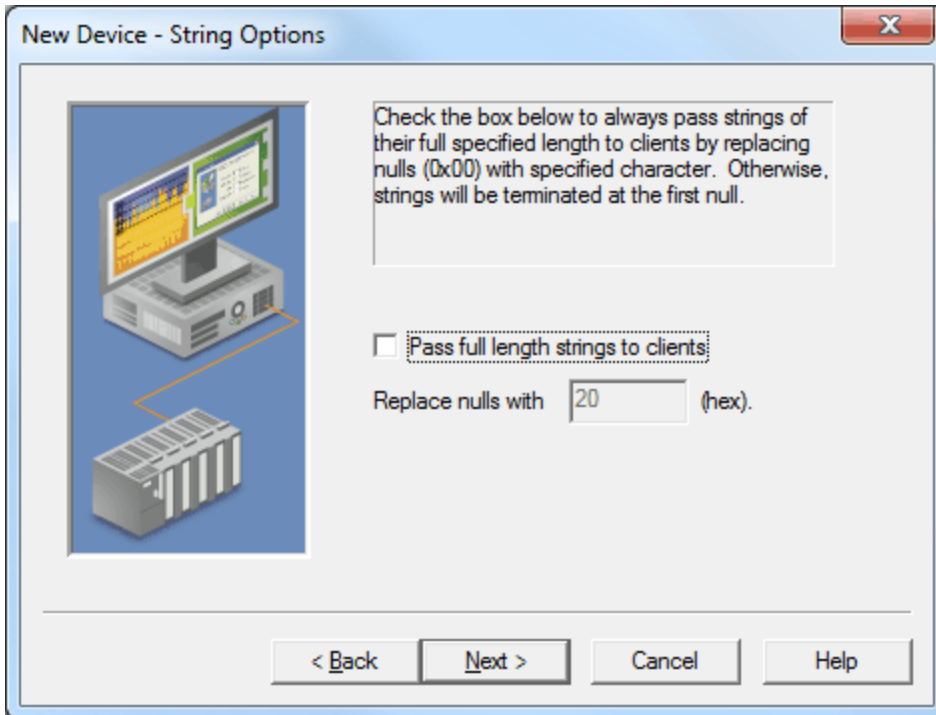
Descriptions of the parameters are as follows:

- **Pass full length strings to clients:** When unchecked, strings sent to the clients will be terminated at the first null character (0x00) encountered in the data. If there are no null characters in the data, one will be placed at the end of the passed string. When checked, all of the characters contained in the memory range will be sent to the client applications. All null characters in the data will be replaced with a user-specified character to allow the client application to display the data's full range. The default setting is unchecked.
- **Replace nulls with ____ (hex):** This parameter specifies the replacement character. The number entered must be in hex. A null character will be placed at the end of the string. For example, 20 results in nulls being replaced by spaces, 2A uses the asterisk character, and so forth.

Data Types Description

Data Type	Description
Boolean	Single bit of an 8 bit value
Byte	Unsigned 8 bit value
Word	Unsigned 16 bit value
Short	Signed 16 bit value
DWord	Unsigned 32 bit value
Long	Signed 32 bit value
Float	32 bit floating point value The driver interprets two consecutive registers as a floating-point value by making the second register the high word and the first register the low word.
BCD	Two byte packed BCD Value range is 0-9999. Behavior is undefined for values beyond this range.
LBCD	Four byte packed BCD Value range is 0-99999999. Behavior is undefined for values beyond this range.
String	Null terminated ASCII string

*For more information, refer to **Address Descriptions**.

Address Descriptions

Address specifications vary depending on the device in use. When attempting to access a data element that goes beyond the range of the device, the server will generate an error message. The default data types for dynamically defined tags are shown in **bold**.

Address Type	Range	Type	Arrays	Access
Discrete Inputs	I0.b-I255.b .b is Bit Number 0-7	**Boolean**	No	Read Only
	IB0-IB255	**Byte**	Yes	Read Only
	IW0-IW254	**Word**, Short	Yes	Read Only
	ID0-ID252	DWord, Long	Yes	Read Only
Discrete Inputs	E0.b-E255.b .b is Bit Number 0-7	**Boolean**	No	Read Only
	EB0-EB255	**Byte**	Yes	Read Only
	EW0-EW254	**Word**, Short	Yes	Read Only
Note: I and E access the same memory area.	ED0-ED252	**DWord**, Long	Yes	Read Only
Discrete Outputs	Q0.b-Q255.b .b is Bit Number 0-7	**Boolean**	No	Read Only
	QB0-QB255	**Byte**	Yes	Read Only
	QW0-QW254	**Word**, Short	Yes	Read Only
	QD0-QD252	**DWord**, Long	Yes	Read Only
Discrete Outputs	A0.b-A255.b .b is Bit Number 0-7	**Boolean**	No	Read Only
	AB0-AB255	**Byte**	Yes	Read Only
	AW0-AW254	**Word**, Short	Yes	Read Only
Note: Q and A access the same memory area.	AD0-AD252	**DWord**, Long	Yes	Read Only
Internal Memory	M0.b-M255.b .b is Bit Number 0-7	**Boolean**	No	Read Only
	MB0-MB255	**Byte**	Yes	Read Only
	MW0-MW254	**Word**, Short, BCD	Yes	Read Only
Note: F and M access the same memory area.	MD0-MD252	**DWord**, Long, LBCD	Yes	Read Only
Data Block Boolean	DBn:KM0.b-KM255.b n: is Block Number (1-255) .b is Bit Number 0-15	**Boolean**	No	Read/Write
Data Block Left Byte	DBn:KL0-KL255 n: is Block Number (1-255)	**Byte**	No	Read/Write
Data Block Right Byte	DBn:KR0-KR255 n: is Block Number (1-255)	**Byte**	No	Read/Write
Data Block Unsigned Word	DBn:KH0-KH255 n: is Block Number (1-255)	**Word**, Short, BCD	Yes	Read/Write
Data Block	DBn:KF0-KF255	**Short**, Word, BCD	Yes	Read/Write

Signed Word	*n:* is Block Number (1-255)			
Data Block Signed Long	DB*n:*KD0-KD254 *n:* is Block Number (1-255)	**Long**, DWord, LBCD	Yes	Read/Write
Data Block Float	DB*n:*KG0-KG254 *n:* is Block Number (1-255)	**Float**	Yes	Read/Write
Data Block String	DB*n::*KS0.l-KS255.l *n:* is Block Number (1-255) .l is String Length 2-128	**String**	No	Read/Write
Data Block Timer	DB*n:*KT0-KT255 *n:* is Block Number (1-255)	**Long**	Yes	Read/Write
Data Block Counter	DB*n:*KC0-KC255 *n:* is Block Number (1-255)	**Word**, Short, BCD	Yes	Read/Write
Extended Data Block Boolean	DX*n:*KM0.b-KM255.b *n:* is Block Number (1-255) .b is Bit Number 0-15	**Boolean**	No	Read/Write
Extended Data Block Left Byte	DX*n:*KL0-KL255 *n:* is Block Number (1-255)	**Byte**	No	Read/Write
Extended Data Block Right Byte	DX*n:*KR0-KR255 *n:* is Block Number (1-255)	**Byte**	No	Read/Write
Extended Data Block Unsigned Word	DX*n:*KH0-KH255 *n:* is Block Number (1-255)	**Word**, Short, BCD	Yes	Read/Write
Extended Data Block Signed Word	DX*n:*KF0-KF255 *n:* is Block Number (1-255)	**Short**, Word, BCD	Yes	Read/Write
Extended Data Block Signed Long	DX*n:*KD0-KD254 *n:* is Block Number (1-255)	**Long**, DWord, LBCD	Yes	Read/Write
Extended Data Block Float	DX*n:*KG0-KG254 *n:* is Block Number (1-255)	**Float**	Yes	Read/Write
Extended Data Block String	DX*n::*KS0.l-KS255.l *n:* is Block Number (1-255) .l is String Length 2-128	**String**	No	Read/Write
Extended Data Block Timer	DX*n:*KT0-KT255 *n:* is Block Number (1-255)	**Long**	Yes	Read/Write
Extended Data Block Counter	DX*n:*KC0-KC255 *n:* is Block Number (1-255)	**Word**, Short, BCD	Yes	Read/Write
Timer Current Values	T0-T255	**Long**	Yes	Read Only
Counter Current Values	C0-C255	**Word**, Short	Yes	Read Only
Counter Current Values	Z0-Z255	**Word**, Short	Yes	Read Only

Note: All offsets for memory types I, Q, and F represent a byte starting location within the specified memory type.

Examples

1. To access bit 3 of Internal Memory F20, declare an address as follows:
F20.3

2. To access Data Block 5 as word memory at element 30, declare an address as follows:
DB5:KH30

3. To access Data Block 2 element 20 and bit 7, declare an address as follows:
DB2:KM20.7

4. To access Data Block 1 as left byte memory at element 10, declare an address as follows:
DB1:KL10

5. To access Internal Memory F20 as a DWORD, declare an address as follows:
FD20

6. To access Input Memory I10 as a Word, declare an address as follows:
IW10

Note: Use caution when modifying Word, Short, DWord, and Long types. For I, Q and F each address starts at a byte offset within the device. Therefore, Words FW0 and FW1 overlap at byte 1. Writing to FW0 will also modify the value held in FW1. Similarly, DWord and Long types can also overlap. It is recommended that these memory types be used so that overlapping does not occur. When using DWords, for example, users can utilize FD0, FD4, FD8 ... and so on to prevent overlapping bytes.

Timers
The Siemens S5 3964R Driver automatically scales T and KT values based on the Siemens S5 time format. The value returned for either a T or KT memory type will already be scaled using the appropriate Siemens time base. As a result, the values are always returned as a count of milliseconds. When writing to T or KT memory types, the Siemens time base will also be applied. To write a value to a timer in the controller, simply write the desired value as a count of milliseconds to the appropriate timer.

Strings
String data is stored in data block registers; thus, the actual number of bytes used to store the data is an even number. For example, if a string of length 5 is specified by DB11:KS1.5, then 3 registers (6 bytes) will be used to store the string data. When writing strings shorter than the maximum specified length (5), a null terminator (0x00) will be added to the end of the string. When strings are read, the full range of registers (3) are read also. Use of string tags with overlapping address ranges should be avoided due to the effects of the null terminators. For more information on how strings can be formatted, refer to **String Options**.

Arrays
Arrays are supported for the memory types indicated in the table above. An array can be declared using the following syntax:

<address> [rows] [cols]
<address> [cols] with an assumed row count of 1.

The maximum size of an array is 128 bytes, where the size of an array is calculated as follows:
size = rows * cols * (data type size in bytes).

The data type size in bytes is 1 for Byte, 2 for Word and Short and 4 for DWord, Long and Float. Timers are an exception, because in that case a data size of 2 bytes should be used.

All locations referenced by an array must exist in the device. If this is not the case, the device will indicate an invalid address upon read or write and the driver will deactivate the tag. For example, if data block 20 has a size of 10 words (KH0 to HK9), then:

1. DB20:KH1 [4] is valid. Element 1 references KH1, element 2 references KH2, element 3 references KH3 and element 4 references KH4.

2. DB20:KG1 [4] would be valid. Element 1 references KH1 and KH2, element 2 references KH3 and KH4, element 3 references KH5 and KH6 and element 4 references KH7 and KH8.

3. DB20:KH8 [4] is invalid. Element 1 references KH8, element 2 references KH9, element 3 references KH10 and element 4 references KH11.

Note: The last two elements reference nonexistent locations.

Counter addresses range from C0 to C255. Therefore C1 [4] is valid. C253 [4] is invalid because the last element references the nonexistent counter C256.

Error Descriptions

The following error/warning messages may be generated. Click on the link for a description of the message.

Address Validation
Address '<address>' is out of range for the specified device or register
Array size is out of range for address '<address>'
Array support is not available for the specified address: '<address>'
Data Type '<type>' is not valid for device address '<address>'
Device address '<address>' contains a syntax error
Device address '<address>' is not supported by model '<model name>'
Missing address

Serial Communications
Communications error on '<channel name>' [<error mask>]
COMn does not exist
COMn is in use by another application
Error opening COMn
Unable to set comm parameters on COMn

Device Status Messages
Device '<device name>' is not responding
Unable to write to '<address>' on device '<device name>'

Driver Warning Messages
Bad block stating at '<address>' for a length of '<bytes>' on device '<device name>'. The block has been deactivated
Initialization conflict error
<Read or Write> request returned error code '<error code hex (decimal)>' for address '<address>' on device '<device name>'

Address Validation

The following error/warning messages may be generated. Click on the link for a description of the message.

Address Validation
Address '<address>' is out of range for the specified device or register
Array size is out of range for address '<address>'
Array support is not available for the specified address: '<address>'
Data Type '<type>' is not valid for device address '<address>'
Device address '<address>' contains a syntax error
Device address '<address>' is not supported by model '<model name>'
Missing address

Address '<address>' is out of range for the specified device or register

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically references a location that is beyond the range of supported locations for the device.

Solution:
Verify that the address is correct; if it is not, re-enter it in the client application.

Array size is out of range for address '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically is requesting an array size that is too large for the address type or block size of the driver.

Solution:
Re-enter the address in the client application to specify a smaller value for the array or a different starting point.

Array support is not available for the specified address: '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically contains an array reference for an address type that doesn't support arrays.

Solution:
Re-enter the address in the client application to remove the array reference or correct the address type.

Data Type '<type>' is not valid for device address '<address>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically has been assigned an invalid data type.

Solution:
Modify the requested data type in the client application.

Device address '<address>' contains a syntax error

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically via DDE contains one or more invalid characters.

Solution:
Re-enter the address in the client application.

Device address '<address>' is not supported by model '<model name>'

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically references a location that is valid for the communications protocol but not supported by the target device.

Solution:
Verify that the address is correct; if it is not, re-enter it in the client application. Also verify that the selected model name for the device is correct.

Missing address

Error Type:
Warning

Possible Cause:
A tag address that has been specified dynamically has no length.

Solution:
Re-enter the address in the client application.

Serial Communications

The following error/warning messages may be generated. Click on the link for a description of the message.

Serial Communications
Communications error on '<channel name>' [<error mask>]
COMn does not exist
COMn is in use by another application
Error opening COMn
Unable to set comm parameters on COMn

Communications error on '<channel name>' [<error mask>]

Error Type:
Serious

Error Mask Definitions:
B = Hardware break detected.
F = Framing error.
E = I/O error.
O = Character buffer overrun.
R = RX buffer overrun.
P = Received byte parity error.
T = TX buffer full.

Possible Cause:
1. The serial connection between the device and the Host PC is bad.
2. The communications parameters for the serial connection are incorrect.

Solution:
1. Verify the cabling between the PC and the device.
2. Verify that the specified communications parameters match those of the device.

COMn does not exist

Error Type:
Fatal

Possible Cause:
The specified COM port is not present on the target computer.

Solution:
Verify that the proper COM port has been selected.

COMn is in use by another application

Error Type:
Fatal

Possible Cause:
The serial port assigned to a device is being used by another application.

Solution:
Verify that the correct port has been assigned to the channel.

Error opening COMn

Error Type:
Fatal

Possible Cause:
The specified COM port could not be opened due an internal hardware or software problem on the target computer.

Solution:
Verify that the COM port is functional and may be accessed by other Windows applications.

Unable to set comm parameters on COMn

Error Type:
Fatal

Possible Cause:
The serial parameters for the specified COM port are not valid.

Solution:
Verify the serial parameters and make any necessary changes.

Device Status Messages

The following error/warning messages may be generated. Click on the link for a description of the message.

Device Status Messages
Device '<device name>' is not responding
Unable to write to '<address>' on device '<device name>'

Device '<device name>' is not responding

Error Type:
Serious

Possible Cause:
1. The serial connection between the device and the Host PC is broken.
2. The communications parameters for the serial connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.
4. The response from the device took longer to receive than the amount of time specified in the "Request Timeout" device setting.

Solution:
1. Verify the cabling between the PC and the device.
2. Verify the specified communications parameters match those of the device.
3. Verify that the Network ID given to the named device matches that of the actual device.
4. Increase the Request Timeout setting so that the entire response can be handled.

Unable to write to '<address>' on device '<device name>'

Error Type:
Serious

Possible Cause:
1. The serial connection between the device and the Host PC is broken.
2. The communications parameters for the serial connection are incorrect.
3. The named device may have been assigned an incorrect Network ID.

Solution:
1. Verify the cabling between the PC and the device.
2. Verify the specified communications parameters match those of the device.
3. Verify that the Network ID given to the named device matches that of the actual device.

Driver Warning Messages

The following error/warning messages may be generated. Click on the link for a description of the message.

Driver Warning Messages
Bad block stating at '<address>' for a length of '<bytes>' on device '<device name>'. The block has been deactivated
Initialization conflict error
<Read or Write> request returned error code '<error code hex (decimal)>' for address '<address>' on device '<device name>'

Bad block starting at '<address>' for a length of '<bytes>' on device '<device name>'. The block has been deactivated

Error Type:
Serious

Possible Cause:
The device has been configured in such a way as to make one or more of the addresses within the block inaccessible.

Solution:
1. Use different addresses.
2. Alter the device configuration.

Initialization conflict error

Error Type:
Warning

Possible Cause:
The device made an attempt to send the driver an unsolicited message at the same time the driver is making a read or write request. This indicates that the device may not be properly configured as the "low priority partner", or is programmed to send unsolicited data. This message may also result from the normal repetition of a packet by the device due to a transient communication problem. Occasional instances of this warning are not necessarily cause for concern.

Solution:
Make sure that the device is configured as the "low priority partner" and is not programmed to send unsolicited data.

<Read or Write> request returned error code '<error code hex (decimal)>' for address '<address>' on device '<device name>'

Error Type:
Serious

Possible Cause:
Consult Siemens RK512 computer link documentation for meaning of "REATEL" error code.

Solution:
Consult Siemens RK512 computer link documentation for meaning of "REATEL" error code.

Note:
By allowing the device to timeout, the driver resolves an "Out of Sync" condition (REATEL code 0x36). Thus, it is normal for this message to be followed by a "Device not responding" error.

Index