

SCHNE

Contents

SCHNE Driver	3
Driver specifications	4
Adding a communication driver to your project	5
Configuring the driver's communication settings	5
About driver worksheets	7
<i>Adding and configuring a Standard Driver Sheet</i>	7
<i>Configuring the Main Driver Sheet</i>	10
Checking the Driver Runtime task.....	15
Troubleshooting	15
Revision history.....	19

SCHNE Driver

SCHNE Driver for Ethernet Communication with M340 and Premium Devices (version 1.7, last revised 15 June 2017).

The SCHNE driver enables communication between the Studio system and remote devices using the Modbus protocol, according to the specifications discussed in this document.

This document assumes that you have read the "Development Environment" section in the main Studio documentation.

This document also assumes that you are familiar with the Microsoft Windows XP/Vista/7 environment. If you are not familiar with Windows, then we suggest using the **Help and Support** feature (available from the Windows **Start** menu) as you work through this document.


Driver specifications

This section identifies all of the software and hardware components required to implement communication between the SCHNE driver in Studio and remote devices using the Modbus protocol.

Driver files

The SCHNE driver package comprises the following files, which are automatically installed in the `Drv` folder of the Studio application directory:

- `SCHNE.DLL`: Compiled driver.
- `SCHNE.INI`: Internal driver file. *You must not modify this file.*
- `SCHNE.MSG`: Internal driver file defining error messages for the possible error codes. (These error codes are described in detail in the [Troubleshooting](#) section.) *You must not modify this file.*
- `SCHNE.PDF`: This document, which provides complete information about using the driver.

 **Note:** You must use a compatible PDF reader to view the `SCHNE.PDF` file. You can install Acrobat Reader from the Studio installation CD, or you can download it from [Adobe's website](#).

You can use the SCHNE driver on the following operating systems:

- Windows XP/Vista/7/8/2008/2012
- Windows CE
- Windows Server 2003/2008

Device specifications

To establish communication, your target device must meet the following specifications:

- Manufacturer: M340 and Premium or compatible devices using the Modbus protocol for Ethernet communication
- Compatible Equipment: M340 and Premium devices or any compatible device with the Modbus protocol
- Programmer Software:

The SCHNE driver supports the following device registers:

Network specifications

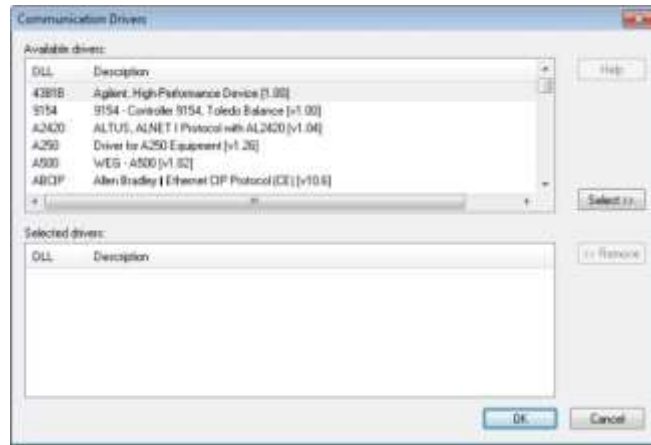
To establish communication, your device network must meet the following specifications:

- Device Communication Port: Modbus protocol default port (502)
- Physical Protocol: Ethernet
- Logic Protocol: Modbus
- Device Runtime Software: None
- Specific PC Board: None
- Cable Wiring Scheme: Regular Ethernet cable

Adding a communication driver to your project

This section explains how to add a communication driver to your project.

1. On the **Insert** tab of the ribbon, in the **Communication** group, click **Add/Remove Driver**.
The *Communication Drivers* dialog is displayed.



Communication Drivers dialog

2. In the *Available drivers* list, click the communication driver that you want to add.
3. Click **Select**.
The driver is added to the *Selected drivers* list.
4. Click **OK**.
The *Communication Drivers* dialog is closed and the selected driver is inserted in the **Drivers** folder in the Project Explorer.

Configuring the driver's communication settings

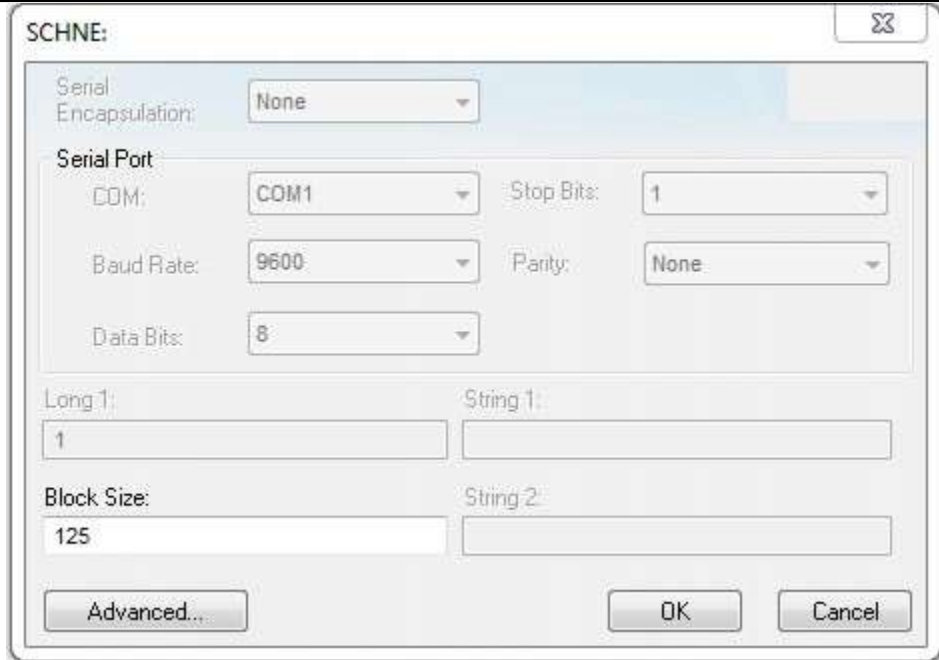
This section explains how to configure the communication settings for the driver.

You must add the communication driver to your project before you can configure its settings. For more information, see [Adding a communication driver to your project](#) on page 6.

The general procedure for configuring a driver's communication settings is the same for all drivers. However, the specific settings are different for each driver, depending on the options and protocols used by the target device.

To configure the communication settings:

1. In the **Comm** tab of the Project Explorer, expand the **Drivers** folder.
The folder contains the drivers that are currently enabled. If you do not see the driver that you want to configure, then you need to add it.
2. Right-click the driver that you want to configure, and then click **Settings** on the shortcut menu.
The *Communication Settings* dialog is displayed.



Communication Settings: SCHNE dialog

3. Configure the remaining, driver-specific settings as needed.

Driver-specific communication settings

Setting	Default Value	Valid Values	Description
Blocksize	64	0 to 125	The size of a block of data that can be transmitted based on the processing capability of the device used. It is 64 (Words) by default for Modicon devices. For Coils (%M) it would be 16 times this number configured(word size). So the maximum block size for coils is 2000 (125*16).

4. Click **OK**.

The settings are saved and the *Communication Settings* dialog is closed.

About driver worksheets

Like the other parts of your project, communication with remote devices is controlled by worksheets. This section explains how to add worksheets to your project and then configure them to associate project tags with device registers.

Each selected driver includes a Main Driver Sheet (MDS) and one or more Standard Driver Sheets (SDS). The Main Driver Sheet is used to define tag/register associations and driver parameters that are in effect at all times, regardless of project behavior. In contrast, Standard Driver Sheets can be inserted to define tag/register associations that are triggered by specific project behaviors.

The configuration of these worksheets is described in detail in the "Communication" chapter of the *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.


For the purposes of this document, only SCHNE driver-specific parameters and procedures are discussed here.

Adding and configuring a Standard Driver Sheet

By default, a communication driver does not include any Standard Driver Sheets. This section explains how to add a Standard Driver Sheet to your project and then configure it.

The SCHNE driver must be added to the project before you can configure any of its worksheets. For more information, see [Adding a communication driver to your project](#) on page 6.

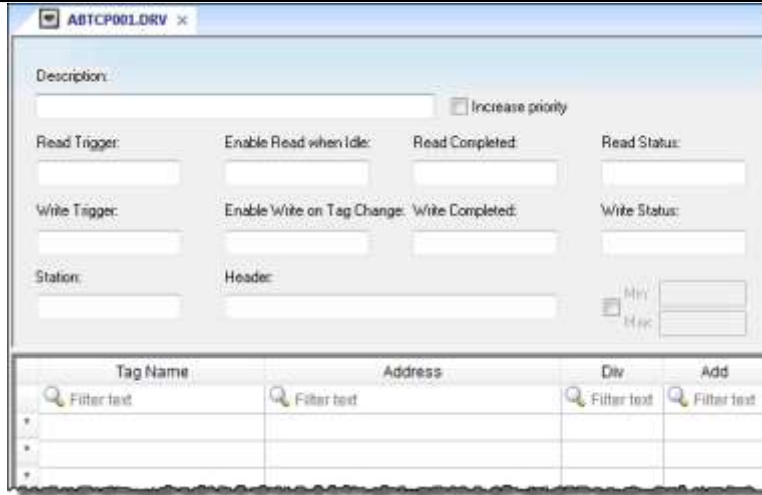
Standard Driver Sheets can be inserted to define additional tag/register associations that are triggered by specific project behaviors.

 **Note:** Most of the settings on this worksheet are standard for all drivers; for more information about configuring these settings, see the "Communication" chapter of the *Technical Reference Manual*. The **Station** and **I/O Address** fields, however, use syntax that is specific to the SCHNE driver.


1. Do one of the following.

- On the **Insert** tab of the ribbon, in the **Communication** group, click **Driver Sheet** and then select **SCHNE** from the list.
- In the **Comm** tab of the Project Explorer, right-click the **SCHNE** folder and click **Insert** on the shortcut menu.

A new SCHNE driver worksheet is inserted into the **SCHNE** folder, and then it is automatically opened for configuring.



Standard Driver Sheet

 **Note:** Worksheets are numbered in order of creation, so the first worksheet is SCHNE001.drv.

2. Configure the Station and Header fields as described below.

Station

Specify the station in the driver sheet using the following syntax. Note that this field cannot be left empty. Examples are given in the table below.

<IP Address>:<SlaveID>:<PortNumber> Where:

<IP Address> is the IP Address of the device on the Ethernet network.

<SlaveID> is the identification number of the Slave device which is default to 1. (This is an optional parameter)

<Port Number> is the port number for the Modbus TCP protocol which is default to 502. (This is an optional parameter)

You can also specify an indirect tag (e.g. {Station}), but the tag that is referenced must follow the same syntax and contain a valid value.

Station Formats

Station	Examples
<IP Address>[:<SlaveID>[:<PortNumber>]]	192.168.110.101 192.168.110.101:1 192.168.110.101:1:502

Header

Specify the address of the first register /coil of a block of registers /coils on the target device. The addresses declared in the body of the worksheet are simply offsets of this **Header** address. When Read and Write actions are executed for the entire worksheet (using **Read Trigger** and **Write Trigger**, respectively), it scans the entire block of registers/coils from the first address to the last.

The **Header** field uses the following syntax:

<Type>:Address Reference

Where: **Type**

Register type - Coil / Holding Register. Valid values are:

- %M - Coils
- %MW - Holding Register

Address Reference

Initial address of the configured type.

After you edit the **Header** field, the development application checks the syntax to determine if it is valid. If the syntax is invalid, then the development application automatically inserts a default value of %M0.

You can also specify an indirect tag (e.g. {MyHeader}), but the tag that is referenced must follow the same syntax and contain a valid value.

Header Format Examples

Header
%M
%MW
%M:10
%MW:100


3. For each tag/register association that you want to create, insert a row in the worksheet body and then configure the row's fields as described below.

Tag Name

Type the name of the project tag.

Address

Specify the address of the associated device register /coil. The valid range of offset varies depending on the type of device used.

 **Note:** Each Standard Driver Sheet can have up to 4096 rows. However, the **Read Trigger**, **Enable Read When Idle**, and **Write Trigger** commands attempt to communicate the entire block of addresses that is configured in the sheet, so if the block of addresses is larger than the maximum block size that is supported by the driver protocol, then you will receive a communication error (e.g., "invalid block size") during run time. Therefore, the maximum block size imposes a practical limit on the number of rows in the sheet.

For examples of how device registers are specified using **Header** and **Address**, see the following table.

Examples of Header and Address fields in Standard Driver Sheet

Header	Address on Studio	Data Type	Description
%MW	W10	INT	Signed 16 bits integer
%MW	W10.2	BOOL	Bit position of a register. Valid values from 0 - 15
%MW	W10.B0, W10.B1	INT	Individual bytes of a register. B0 gets the least significant byte and B1 gets the MSB of a register.
%MW	F100	REAL	32 bit Real variables
%MW	UW20	UWORD	Unsigned 16 bit word
%MW	DW30	DWORD	Signed 32 bit DWORD / DINT
%MW	UDW40	UDWORD	Unsigned 32 bit DWORD
%MW	S50.10	string[10]	String of size 10
%MW	DATE100	DATE	32-bit DATE type of format YYYY-MM-DD.
%MW	TIME10	TIME	TIME type in milliseconds
%MW	TOD300	TOD	Time Of Day in HH:MM:SS format. Valid Range is 00:00:00 to 23:59:59
%MW	DT20	DT	DateTime in YYYY-MM-DD-HH:MM:SS. Valid range of Year is 1990 to 2100
%M	10	EBOOL	This is a Coil Address.
%M	21	EBOOL	This is a Coil Address.

For more information about the device registers and addressing, please consult the manufacturer's documentation.


4. Save and close the worksheet.

Configuring the Main Driver Sheet

When you add the SCHNE driver to your project, the Main Driver Sheet is automatically included in the **SCHNE** folder in the Project Explorer. This section describes how to configure the worksheet.

The SCHNE driver must be added to the project before you can configure any of its worksheets. For more information, see [Adding a communication driver to your project](#) on page 6.

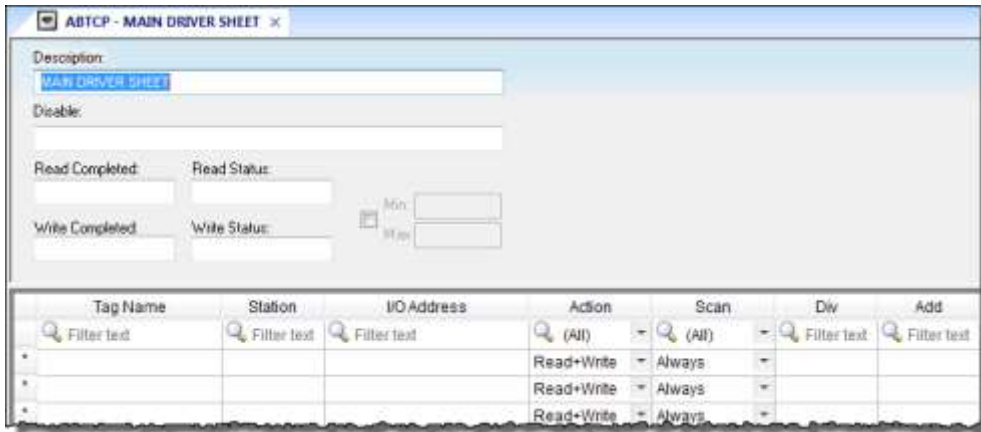
The Main Driver Sheet is used to define tag/register associations and driver parameters that are in effect at all times, regardless of project behavior. The worksheet is continuously processed during project runtime.

 **Note:** Most of the settings on this worksheet are standard for all drivers; for more information about configuring these settings, see the “Communication” chapter of the *Technical Reference Manual*. The **Station** and **I/O Address** fields, however, use syntax that is specific to the SCHNE driver.

1. Do one of the following.

- On the **Insert** tab of the ribbon, in the **Communication** group, click **Main Driver Sheet** and then select **SCHNE** from the list.
- In the **Comm** tab of the Project Explorer, expand the **SCHNE** folder and then double-click **MAIN DRIVER SHEET**.

The Main Driver Sheet is displayed.



Main Driver Sheet

2. For each tag/register association that you want to create, insert a row in the worksheet body and then configure the row's fields as described below.

Tag Name

Type the name of the project tag.

Station

Specify the station in the MDS on every line. Note that this field cannot be left empty. Examples are given in the table below.

- For communicating with memory addresses, use the following syntax.

<IP Address>:<SlaveID>:<PortNumber> Where:

<IP Address> is the IP Address of the device on the Ethernet network.

<SlaveID> is the identification number of the Slave device which is default to 1. (This is an optional parameter)

<Port Number> is the port number for the Modbus TCP protocol which is default to 502. (This is an optional parameter)

You can also specify an indirect tag (e.g. {Station}), but the tag that is referenced must follow the same syntax and contain a valid value.

- For communicating with TAG Header, use the following syntax.

[XSY File:PLCTYPE:]<IP Address>:<SlaveID>:<PortNumber> Where:

[XSY File] is the name of the XSY file containing the tags. If just the name of the file is given, the file should be located in the Project folder. Else, the file name should contain the full path of the location of the file.

[PLC Type] is the type of the PLC used. The following are valid values for PLCType.

- M340
- Premium

Station Formats

Station	Examples
[XSY File:PLCTYPE:]<IP Address>:<SlaveID>:<PortNumber>	Unity_Variable_Export.XSY:M340:192.168.110.101

Station	Examples
	C:\Unity_Variable_Export.XSY:M340:192.168.110.101 Unity_Variable_Export.XSY:M340:192.168.110.101:1 "C:\Unity_Variable_Export.XSY":M340:192.168.110.101:1:502 Unity_Variable_Export.XSY:Premium:192.168.110.101:1:502
<IP Address>[:SlaveID:[PortNumber]]	192.168.110.101 192.168.110.101:1 192.168.110.101:1:502

I/O Address

Specify the address of the associated device register or coil. Communication can be done in the Main Driver Sheet using Tag Names (Only on MDS) or using Memory Addresses For communicating with TAG Headers , use the following syntax:

<TagName>

For all memory address types use the following syntax:

<Type><Address>.<Bit/Byte/Length>

Where:

<TagName>

The name of the TAG from the XSY file.

<Type>

The register type. Valid values are %MW, %MF, %MDW, %MUDW, %MS, %MDATE, %MTIME, %MTOD, %MDT for Holding Registers and %M for Coils

<Address>


The address of the device register or coil.


<Bit/Byte/Length> (Optional)

This parameter is only used for Holding Registers. This can denote a bit number or a byte number for data types W, DW and UDW or the length of a string for String type addresses. The Bit number could be a valid bit number of the register, the byte number is the specific byte of a word and the length is the length of the string to be read. Note that this parameter is not valid on coils.

For addressing Tags of single and multi-dimensional Array types, the following syntaxes are valid.


- Array_1D[index], Array_2D[index1, index2], Array_3D[index1, index2, index3] and so on
- Array_1D[index], Array_2D[index1][index2], Array_3D[index1][index2][index3] and so on


 **Note:** The driver supports array tags of any dimension, including array types of any dimension. However, arrays of array types is currently not supported.


 **Note:** When using TAG types addresses, make sure to use the correct PLC type in the station field. The PLC type determines the memory alignment and choosing the wrong type might cause unexpected results


I/O Address Format

I/O Address Type	Example
TAG Names	Sys_Excep_Reset Sys_Excep Struct3.Struct2.STRING_15 Struct4.Struct2Array[1].WORD_1 Array1Test[1] Array2Test_2D[1][2], Array2Test_2D[1,2] Array3Test_3D[1][2][3], Array3Test_3D[1,2,3]
Memory Address	%MW100 %MW100.B0, %MW100.B1 (Byte of a Word) %MW100.12 (Individual Bit of a Word) %MF10 (Real format) %MDW10 (DWORD) %MUDW10 (unsigned DWORD) %MDATE200 %MTIME30 %MTOD300 %MDT20 %M100, %M1 (Coils)


 **Note:** When exporting the variables and definitions to an XSY file on the Unity Pro software, some of the DDTs do not get exported. These are those that are predefined on the Unity Pro Library and are considered "Protected". So variables created with such DDTs do not work. Hence, variables are to be created only with the Application DDTs, the ones in yellow on the DDT Table. These are always exported to the XSY file and hence will work.

 **Note:** Also, note that coils which are configured as EBOOL on the programming software are prohibited in DDTs. So they cannot be used in them. However Arrays are valid.

 **Note:** The TIME data type on the Studio is displayed as a number and is read in milliseconds. When the time is given in milliseconds on the Studio, the Unity Pro converts it automatically and displays in its own format.

 **Note:** The range of valid values for the TOD data type is from 00:00:00 to 23:59:59

The range of valid values for the YEAR in DT data type is from 1990 to 2100, the month and day as in the Calendar days and the Time range from 00:00:00 to 23:59:59

 **Note:** The Main Driver Sheet can have up to 32767 rows. If you need to configure more than 32767 communication addresses, then either configure additional Standard Driver Sheets or create additional instances of the driver.

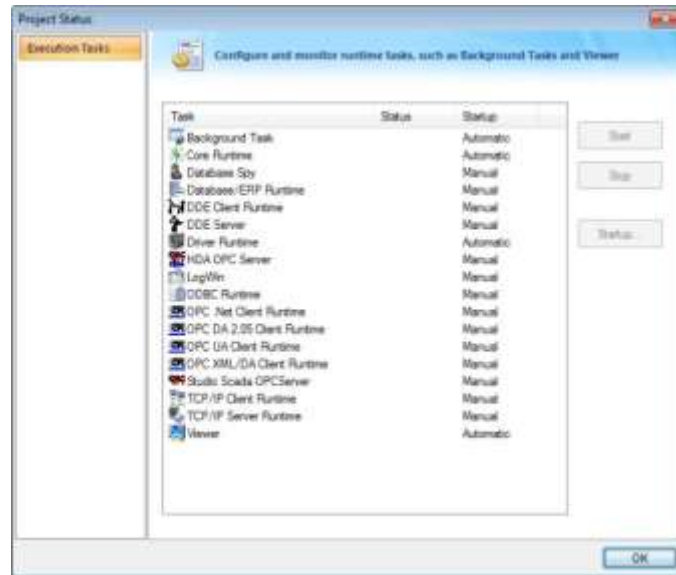
3. Save and close the worksheet.

Checking the Driver Runtime task

This section describes how to check the status of the Driver Runtime task in the list of execution tasks.

The Driver Runtime task handles communication with remote devices and the processing of the driver worksheets. By default, the task is configured to start up automatically when the project is run, but you can check it for yourself.

1. On the **Home** tab of the ribbon, in either the **Local Management** or the **Remote Management** group (depending on where you project server will be running), click **Tasks**.
The *Project Status* dialog is displayed.



Project Status dialog

2. Verify that the **Driver Runtime** task is set to **Automatic**.
 - If the setting is correct, then proceed to the next step.
 - If the **Driver Runtime** task is set to **Manual**, select the task and then click **Startup** to change the task to **Automatic**.
3. Click **OK** to close the *Project Status* dialog.

Troubleshooting

This section lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.


Checking status codes

If the SCHNE driver fails to communicate with the target device, then the database tag(s) that you configured for the **Read Status** and **Write Status** fields of the driver sheets will receive a status code. Use this status code and the following tables to identify what kind of failure occurred and how it might be resolved.

Status codes for the driver

Error	Description	Possible Causes	Procedure To Solve
0	OK	No problem detected	None
1	Error Sending Buffer	Unable to send the command to PLC	Check the connection and try again
2	Received wrong Station in response	The response received is not from a valid station.	Check to make sure that the station is configured correctly.
3	Invalid CRC	The CRC received does not match with calculated CRC.	None.
4	Invalid Sequence number in response	The response sequence number does not match that of the sent sequence number	Try sending the command once again, else restart the driver
6	Write Command is not supported on this header	The write command is not supported on this header	The header given does not support write command, please use a header which supports it.
7	Received Invalid Function code in response	The received response function code does not match the request function code sent by Driver	Try sending the command once again, else restart the driver
8	Protocol error returned	The PLC returned an error	Check the error code returned by PLC
10	Invalid Tags in XSY File	The Tag specified in the MDS is not a valid tag from the XSY file specified or the format of the tag is incorrect.	Check if the Tag exists in the XSY file mentioned and is given correctly on the MDS.
11	Invalid Block Size	The block size given exceeds the maximum block size allowed on the driver.	Give a valid block size number.
12	For communicating with holding registers (%MW) please use the header %MW	Trying to add an address starting with any data type of the Holding Register when the header on the driver sheet is %M.	When the header is %M the address should just be the number of the coil address only. To access holding registers, %MW header should be used.
13	Error to communicate with the controller. You are using the coil header (%M) to communicate with holding registers (%MW), please modify the header of your standard driver sheet to %MW.	Trying to run the driver, with a sheet which has header configured as coils (%M) and one of the addresses that of a holding register.	Check all the driver sheets and make sure that headers with %M (Coils) do not have the holding register addresses.
14	The Time Of Day value given is out of range or invalid. Please give the correct range in the format HH:MM:SS.	One or more values in the hour, minutes or seconds of the TOD data type is out of range.	Check and give the correct values for Hour, Minute and Second in the correct format HH:MM:SS
15	The DateTime value is invalid. Please give the correct format as YYYYMM-DD-HH:MM:SS.	One or more values in the year, month, day, hour, minutes or seconds of the DT data type is out of range	Check and give the correct values for Year, Month, Day, Hour, Minute and Second in the correct format YYYY-MM-DDHH:MM:SS

Common status codes

Status Code	Description	Possible Causes	Procedure To Solve
0	OK	Communicating without error.	None required.
-15	Timeout waiting for message to start	<ul style="list-style-type: none"> Disconnected cables. PLC is turned off, in stop mode, or in error mode. Wrong station number. Wrong parity (for serial communication). Wrong RTS/CTS configuration (for serial communication). 	<ul style="list-style-type: none"> Check cable wiring. Check the PLC mode — it must be RUN. Check the station number. Increase the timeout in the driver's advanced settings. Check the RTS/CTS configuration (for serial communication).
-33	Invalid driver configuration file	The driver configuration file (<i>drivername.INI</i>) is missing or corrupt.	Reinstall the driver.
-34	Invalid address	The specified address is invalid or out of range.	Check the supported range of addresses described in this document, and then correct the address.
-35	Driver API not initialized	The driver library was not initialized by the driver.	Contact technical support.
-36	Invalid data type	The specified data type is invalid or out of range.	Check the supported data types described in this document, and then correct the data type.
-37	Invalid header	The specified header in the driver worksheet is invalid or out of range.	Check the supported range of headers described in this document, and then correct the header.
-38	Invalid station	The specified station in the driver worksheet is invalid or out of range.	Check the supported station formats and parameters described in this document, and then correct the station.
-39	Invalid block size	Worksheet is configured with a range of addresses greater than the maximum block size.	<p>Check the maximum block size number of registers described in this document, and then configure your driver worksheet to stay within that limit. Keep in mind that you can create additional worksheets.</p> <p> Note: If you receive this error from a Main Driver Sheet or Tag Integration configuration, please contact Technical Support.</p>
-40	Invalid bit write	Writing to a bit using the attempted action is not supported.	<ul style="list-style-type: none"> Writing to a bit using Write Trigger is not supported in some drivers. Modify the driver worksheet to use Write On Tag Change. The bit is read-only.
-42	Invalid bit number	The bit number specified in the address is invalid. The limit for the bit number depends on the registry type.	Check the addresses to see if there are bit numbers configured outside the valid range for the registry.
-43	Invalid byte number	The byte number specified in the address is invalid. The limit for the byte number depends on the registry type.	Check the addresses to see if there are byte numbers configured outside the valid range for the registry.
-44	Invalid byte write	Writing to a byte using the attempted action is not supported.	The byte is read-only or inaccessible.

-45	Invalid string size	The string is more than 1024 characters.	Modify the addresses that have string data type to be less than 1024 characters.
-56	Invalid connection handle	The connection is no longer valid.	Please contact Technical Support.
-57	Message could not be sent	The socket was unable to send the TCP or UDP message.	<ul style="list-style-type: none"> • Check the station IP address and port number. • Confirm that the device is active and accessible. Try to ping the address.
-58	TCP/IP could not send all bytes	The TCP/IP stack was not able to send all bytes to destination.	<ul style="list-style-type: none"> • Check the station IP address, port number and/or ID number. • Confirm that the device is active and accessible. • Try to ping the address.
-60	Error to establish TCP/IP connection	Error while establishing a TCP/IP connection with the slave device. Possibly incorrect IP address or port number in the specified station.	<ul style="list-style-type: none"> • Check the station IP address, port number and/or ID number. • Confirm that the device is active and accessible. • Try to ping the address.
-61	TCP/IP socket error	The TCP/IP connection has been closed by the device.	Confirm that the device is active and accessible. Try to ping the address.
-62	UDP/IP receive call returned error	The UDP socket is in error.	<ul style="list-style-type: none"> • Check the station IP address, port number and/or ID number. • Confirm that the device is active and accessible. • Try to ping the address.
-63	UDP/IP error initializing	The UDP socket initialization failed.	Confirm that the operating system supports UDP sockets.
-64	UDP/IP receive call returned error	The UDP socket is in error.	<ul style="list-style-type: none"> • Check the station IP address, port number and/or ID number. • Confirm that the device is active and accessible. • Try to ping the address.
-65	UDP/IP bind error, port number may already be in use	The driver was not able to bind the UDP port.	<ul style="list-style-type: none"> • Check the port number used by the driver. • Check for other programs that might be bound to the UDP port.

Monitoring device communications

You can monitor communication status by establishing an event log in Studio's *Output* window (LogWin module). To establish a log for Field Read Commands, Field Write Commands and Serial Communication, right-click in the *Output* window and select the desired options from the pop-up menu.

You can also use the LogWin module to establish an event log on a remote unit that runs Windows Embedded. The log is saved on the unit in the `celog.txt` file, which can be downloaded later.

If you are unable to establish communication between Studio and the target device, then try instead to establish communication using the device's own programming software. Quite often, communication is interrupted by a hardware or cable problem or by a device configuration error. If you can successfully communicate using the programming software, then recheck the driver's communication settings in Studio.

Contacting Technical Support

If you must contact Technical Support, please have the following information ready:

- **Operating System** and **Project Information**: To find this information, click **Support** in the **Help** tab of the ribbon.
- **Driver Version** and **Communication Log**: Displays in the *Output* window (LogWin module) when the driver is enabled and the project is running is running.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

Revision history

This section provides a log of all changes made to the driver.

Revision history

Driver Version	Revision Date	Description of Changes	Author
1.0	Sep. 26, 2013	First driver revision	Paulo Balbino/Priya Yennam
1.0	Oct. 4, 2013	Added station syntax documentation. No driver changes.	Priya Yennam
1.1	Jan 22, 2014	<ul style="list-style-type: none"> • Added support for Windows CE. • Added a new data format: DATE 	Priya Yennam
1.2	Feb 03, 2014	Added support for Coils (%M)	Priya Yennam
1.3	Nov 18, 2014	<ul style="list-style-type: none"> • Added note with list of unsupported data types. • Added status code for block size error. 	Priya Yennam
1.4	Mar 05, 2015	<ul style="list-style-type: none"> • Added support to TIME data type. • Modified the %MW and %M headers such that each is used separately as header on driver sheets for Holding Registers and Coils respectively 	Priya Yennam
1.5	Nov 10, 2015	<ul style="list-style-type: none"> • Added support for Time Of Day (TOD) • Added support for DateTime(DT) • Fixed an issue with lower case topological address tags not being read properly with Tag integration 	Priya Yennam/Anushree Phanse
1.6	Feb 16, 2017	<ul style="list-style-type: none"> • Fixed issue of not communicating properly with Bitranks in DDT (BOOL bits of a Word) • Fixed issue of not communicating properly with subsequent members in a DDT when it contains Bitranks (BOOL bits of a WORD) 	Anushree Phanse
1.7	June 15, 2017	•Improved initialization performance	Anushree Phanse