

OMNEP

Driver Guide

Version: 1.00

Contents

OMNEP Driver	3
Driver specifications	3
Driver files	3
Target Display Unit	4
Device specifications	4
Working with Communication Drivers	5
Adding a communication driver to the project.....	5
Configuring the driver communication settings.....	5
About driver worksheets.....	8
Adding and configuring a Standard Driver Sheet	8
Configuring the Main Driver Sheet.....	11
Checking the Communication Driver task.....	13
Supported Registers	15
Troubleshooting	18
Checking status codes	18
Monitoring device communications.....	19
Revision history.....	20
Contacting Technical Support.....	20

CHAPTER 1

OMNEP Driver

The OMNEP Driver for OMRON NJ/NX series EtherNet/IP devices enables communication between the Studio system and remote devices, as per the specifications discussed in this document.

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

We assume you are familiar with working in a Windows environment, and we do not attempt to explain Windows navigation, file management, and so forth. If you are unfamiliar with any of these procedures, we recommend using the Windows Help feature or consulting your Microsoft Windows documentation.

Driver specifications

This section identifies the software and hardware components required to implement communication between the OMNEP driver in Studio and remote devices.

Driver files

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

OMNEP.DLL: Compiled driver.

OMNEP.INI: Internal driver file. *You must not modify this file.*

OMNEP.MSG: Internal driver file defining error messages for error codes. (These error codes are described in detail in the Troubleshooting section.) *You must not modify this file.*

OMNEP.PDF: This document provides information about using the driver.

Note: You must use a compatible PDF reader to view the OMNEP.PDF file. You can install Acrobat Reader from the Studio installation CD, or download it from Adobe's website.

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

- Windows:
 - Windows 11
 - Windows 10, version 1909 or later (including LTSC/LTSB versions)
- Windows Server:
 - Windows Server 2022
 - Windows Server 2019
 - Windows Server 2016
- Windows Embedded:
 - Windows 11 IoT Enterprise
 - Windows 10 IoT Enterprise (LTSC/LTSB version only)

Target Display Unit

To establish communication, the target display unit must be one of the following series:

- IPC series
- PC/AT series

Device specifications

To establish communication, the external device must meet the following specifications:

Series Name	CPU	Link Unit	Interface	Comment
NJ Series	NJ501-0000 NJ301-0000 NJ101-0000	Built-in EtherNet/IP Port (PORT1) on CPU unit	Ethernet	NJ101: Unit version.1.10 or later.
NX1P Series	NX1P2-000000 NX1P2-0000000	Built-in EtherNet/IP Port (PORT1) on CPU unit	Ethernet	Unit version.1.13 or later
NX1 Series	NX102-0000	Built-in EtherNet/IP Port (PORT1) on CPU unit Built-in EtherNet/IP Port (PORT2) on CPU unit	Ethernet	Unit version.1.30 or later
NX5 Series	NX501-1000	Built-in EtherNet/IP Port (PORT1) on CPU unit	Ethernet	
	NX502-1000	Built-in EtherNet/IP Port (PORT1) on CPU unit Built-in EtherNet/IP Port (PORT2) on CPU unit	Ethernet	Unit version.1.60 or later
NX7 Series	NX701-0000	Built-in EtherNet/IP Port (PORT1) on CPU unit Built-in EtherNet/IP Port (PORT2) on CPU unit	Ethernet	Unit version.1.10 or later

CHAPTER 2

Working with Communication Drivers

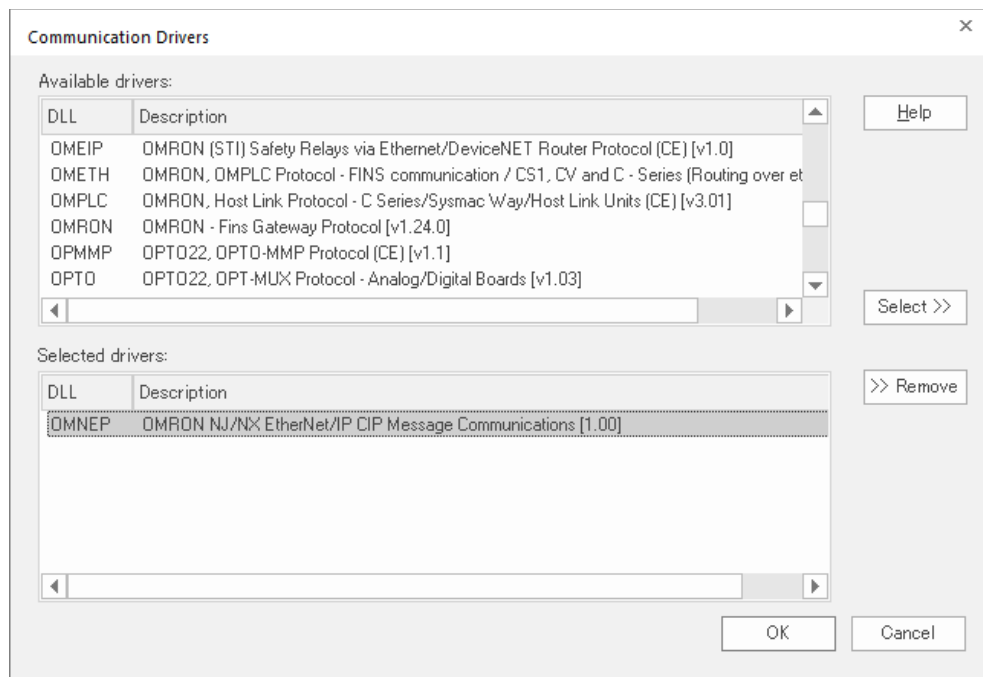
This section explains how to add a communication driver to the project and how to configure the communication settings for the driver.

Adding a communication driver to the project

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

1. On the **Insert** tab of the ribbon, in the **Communication** group, click **Add/Remove Driver**.

The *Communication Drivers* dialog is displayed.



2. In the *Available drivers* list, click the communication driver to add.
3. Click **Select** to add the driver 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 communication settings

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

You must add the communication driver to the project before you can configure its settings. For more information, see *Adding a communication driver to the project* on Page 5.

The general procedure for configuring the driver 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.

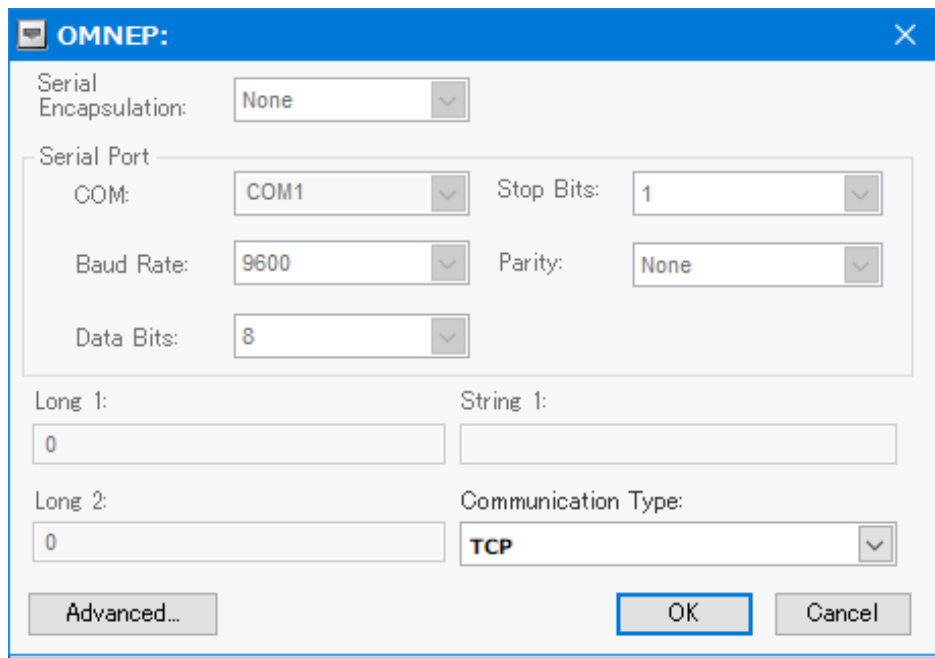
To configure the communication settings:

1. In the Project Explorer's **Comm** tab, 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 it needs to be added.

2. Right-click the driver to configure, and then from the shortcut menu, click **Settings**.

The *Communication Settings* dialog is displayed.

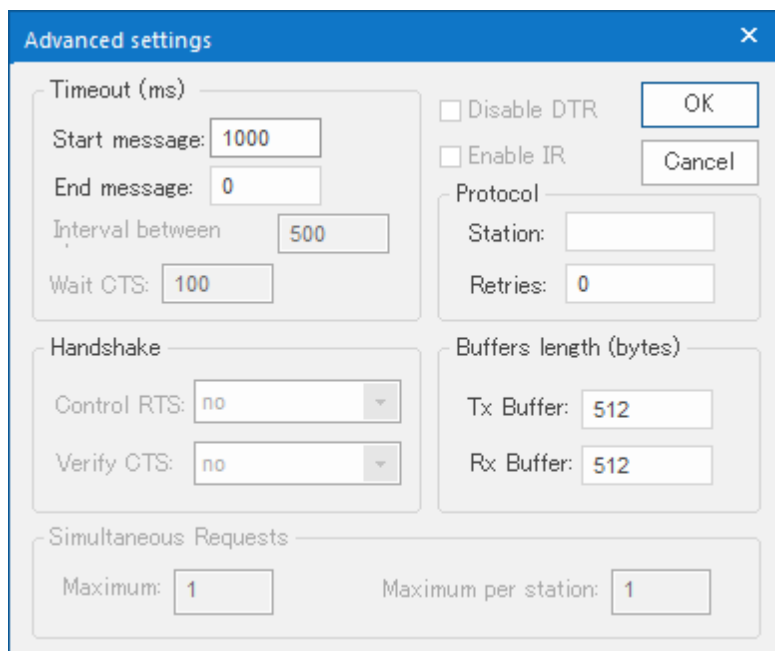
The image shows a dialog box titled "OMNEP:". It contains several configuration options. At the top, "Serial Encapsulation:" is set to "None". Below this, a "Serial Port" section includes "COM:" set to "COM1", "Baud Rate:" set to "9600", "Data Bits:" set to "8", "Stop Bits:" set to "1", and "Parity:" set to "None". Further down, there are two text input fields labeled "Long 1:" and "Long 2:", both containing the value "0". To the right of these is a "String 1:" label and an empty text input field. Below the "Long 2:" field is a "Communication Type:" dropdown menu currently set to "TCP". At the bottom of the dialog are three buttons: "Advanced...", "OK", and "Cancel".

3. Configure the remaining driver-specific settings as required.

Driver-specific communication settings

Setting	Default Value	Valid Values	Descriptions
Communication Type	TCP	<ul style="list-style-type: none">• TCP• UDP	Select the port to connect to the device.

4. Click **Advanced** to display the *Advanced settings* dialog.



The image shows a Windows-style dialog box titled "Advanced settings" with a close button (X) in the top right corner. The dialog is divided into several sections with rounded corners. The "Timeout (ms)" section contains four input fields: "Start message:" with the value 1000, "End message:" with the value 0, "Interval between" with the value 500, and "Wait CTS:" with the value 100. The "Handshake" section contains two dropdown menus: "Control RTS:" set to "no" and "Verify CTS:" set to "no". The "Simultaneous Requests" section contains two input fields: "Maximum:" with the value 1 and "Maximum per station:" with the value 1. On the right side, there are two checkboxes: "Disable DTR" (unchecked) and "Enable IR" (unchecked). Below these are two buttons: "OK" and "Cancel". A "Protocol" section contains two input fields: "Station:" (empty) and "Retries:" with the value 0. The "Buffers length (bytes)" section contains two input fields: "Tx Buffer:" with the value 512 and "Rx Buffer:" with the value 512.

Section	Parameter	Value
Timeout (ms)	Start message:	1000
	End message:	0
	Interval between	500
	Wait CTS:	100
Handshake	Control RTS:	no
	Verify CTS:	no
Simultaneous Requests	Maximum:	1
	Maximum per station:	1
Protocol	Station:	
	Retries:	0
Buffers length (bytes)	Tx Buffer:	512
	Rx Buffer:	512

Note: The settings in this dialog box are standard for all drivers. For more information about configuring these settings, see the "Communication" chapter of the *Help Manual*.

5. Click **OK** to close the *Advanced settings* dialog.
6. Click **OK** to save the settings and close the Communication Settings dialog.

CHAPTER 3

About driver worksheets

Like the other parts of the project, communication with remote devices is controlled with worksheets. This section explains how to add worksheets to the project and 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 defines tag/register associations and driver parameters that are in effect at all times, regardless of project behavior. In contrast, Standard Driver Sheets define tag/register associations that are triggered by specific project behavior.

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

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

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 the project and then how to configure the SDS.

The OMNEP driver must be added to the project before you can configure any of its worksheets. For more information, see *Adding a communication driver to the project* on Page 5.

Insert SDS to define additional tag/register associations triggered by specific project behavior.

Note: Most of the settings in the SDS are standard for all drivers. For more information about configuring these settings, see the "Communication" chapter of the *Help Manual*. The **Station** and **I/O Address** fields, however, use syntax that is specific to the OMNEP driver.

1. Do one of the following.
 - On the **Insert** tab of the ribbon, in the **Communication** group, click **Driver Sheet** and then select **OMNEP** from the list.
 - From the Project Explorer's **Comm** tab, right-click the **OMNEP** folder and from the shortcut menu, click **Insert**.

A new OMNEP driver worksheet is inserted into the **OMNEP** folder and opened for configuring.

Description

☐ Increase priority

Read Trigger: Enable Read when: Read Completed: Read Status:

Write Trigger: Enable Write on Tag: Write Completed: Write Status:

Station: Header: ☐ Min: ☐ Max:

	Tag Name	Address	Div	Add
	Filter text	Filter text	Filter text	Filter text
*				
*				

Note: Worksheets are numbered in order of creation, so the first worksheet is OMNEP001.drv.

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

For more information, see *Station* on Page 9 and *Header* on Page 9.

Note: Station field cannot be left empty.

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

For more information, see *Tag Name* on Page 10 and *Address* on Page 10.

4. Save and close the worksheet.

Station

Specify the station in the driver sheet using the following syntax.

<IP Address>

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

The port number is fixed [UDP: 44818, TCP: 44818].

During runtime, you can specify a tag in curly brackets to change the station (e.g. {Station}), but the tag that is referenced must follow the same syntax and contain a valid value.

Header

Specify the address of the first register of a block of registers 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), the entire block of registers/coils are scanned from the first to the last address.

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

<Variable data type>:(Variable name)

When the header is “blank” and the setting is for an unsupported device type, as there is no default address it is displayed as blank.

When the header is already defined, and the setting is for an unsupported device type or an unsupported range, the header is not updated.

This driver supports Curly Brackets in the Header, the same as other drivers. Set the header with a String variable, and if the string is not a valid address name, the driver sets an “Invalid header” error to the Read or Write Status.

Tag Name

Type the name of the project tag.

Address

Specify an offset from Header address.

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

<Address Offset>.[Bit]

<Address Offset>.[Length] (String Format)

<Address Offset>

This is a parameter added to the (Address Number) parameter configured in the **Header** field.

[Bit] (Optional)

This parameter is the bit number. The String format does not use this parameter.

The bit range depends on the format defined in the header or the device address type (bit length).

- Byte type: 0...7
- Word type: 0...15
- Double word type: 0...31

If the setting value is out of range, the value is not updated.

[Length] (Optional)

This value denotes the length of the string to read from a String type address.

The valid range: 1...1986.

If the setting value is out of range, the value is not updated.

Since String tags are limited to 1024 characters in the Studio system, strings received from the PLC with more than that will be truncated, and the remaining information will be discarded. In addition, the maximum number of characters that can be written to the PLC is also 1024, and the resulting data in the remainder of the larger string in the PLC will be indeterminate.

Note: The tag names are case sensitive. Tags with incorrect spelling or case will result in errors.

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 are configured in the sheet. So, if the block of addresses is larger than the maximum block size supported by the driver protocol, then it will result in a communication error (e.g., "invalid block size") during runtime. Therefore, the maximum block size imposes a practical limit on the number of rows in the sheet.

Note: Communication status does not change due to invalid **Variable Name** and/or **Bit** request.

When the specified PLC variable name or bit number in the driver sheet is invalid (does not exist in the PLC), the value is displayed as "?", but the communication status (**Read Status** / **Write Status**) remains "OK" (i.e. the value remains 0).

Examples of Header and Address

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

Register Address in the Device	Representation	Header Field	Address Field
VariableNameI[10]	Decimal (Word)	INT:VariableNameI	10
VariableNameB[15]	Decimal (Bit)	BOOL:VariableNameB	15

Configuring the Main Driver Sheet

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

The OMNEP driver must be added to the project before you can configure any of its worksheets. For more information, see *Adding a communication driver to the project* on Page 5.

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

Note: Most of the settings in this worksheet are standard for all drivers. For more information about configuring these settings, see the "Communication" chapter of the *Help Manual*. The **Station** and **I/O Address** fields, however, use syntax that is specific to the OMNEP 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 **OMNEP** from the list.
 - ☐ From the Project Explorer's **Comm** tab, expand the **OMNEP** folder and then double-click **MAIN DRIVER SHEET**.

The Main Driver Sheet is displayed.

Tag Name	Station	I/O Address	Action	Scan	Div	Add
Filter text	Filter text	Filter text	(All)	(All)	Filter text	Filter text
*			Read+Write	Always		
*			Read+Write	Always		

- For each tag/register association to create, insert a row in the worksheet body and then configure the row's fields.

For more information, see *Tag Name* on Page 12, and *Station* on Page 12, and *I/O Address* on Page 12.

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.

- Save and close the worksheet.

Tag Name

Type the name of the project tag.

Station

Please refer to information on the Station field for the Standard Driver Sheet. See *Station* on Page 9.

I/O Address

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

The **I/O Address** field uses the following syntax:

<Variable data type>:(Variable name).[Bit]

<Variable data type>:(Variable name).[Length]

String Encode: String data is treated as UTF-8.

<Variable Data Type> / (Variable name)

For information about available register types and valid ranges of each register, see *Supported Registers* on Page 15.

[Bit] / [Length]

Please refer to information on the Address field for the Standard Driver Sheet. See *[Bit] (Optional) / [Length] (Optional)* on Page 10.

Note: Communication status does not change due to invalid **Variable Name** and/or **Bit** request.

When the specified PLC variable name or bit number in the driver sheet is invalid (does not exist in the PLC), the value is displayed as "?", but the communication status (**Read Status** / **Write Status**) remains "OK" (i.e. the value remains 0).

Examples of Specifying Variables

Pattern	Variable Name in BOS	Note
Simple pattern(BOOL)	BOOL:VariableNameB	BOOL type variable "VariableNameB"
Simple pattern(INT)	INT:VariableNameI	INT type variable "VariableNameI"
Structure names are separated by dots.	INT:VariableName.ElementName	Specify the element name "ElementName" of the "VariableName" structure.
Word device and can be accessed as a bit: "Bit" number should be set after variable name.	INT:VariableNameI.5	Use the INT type variable "VariableNameI" as Bit address. Numeric characters cannot be used as the first character of the variable name.

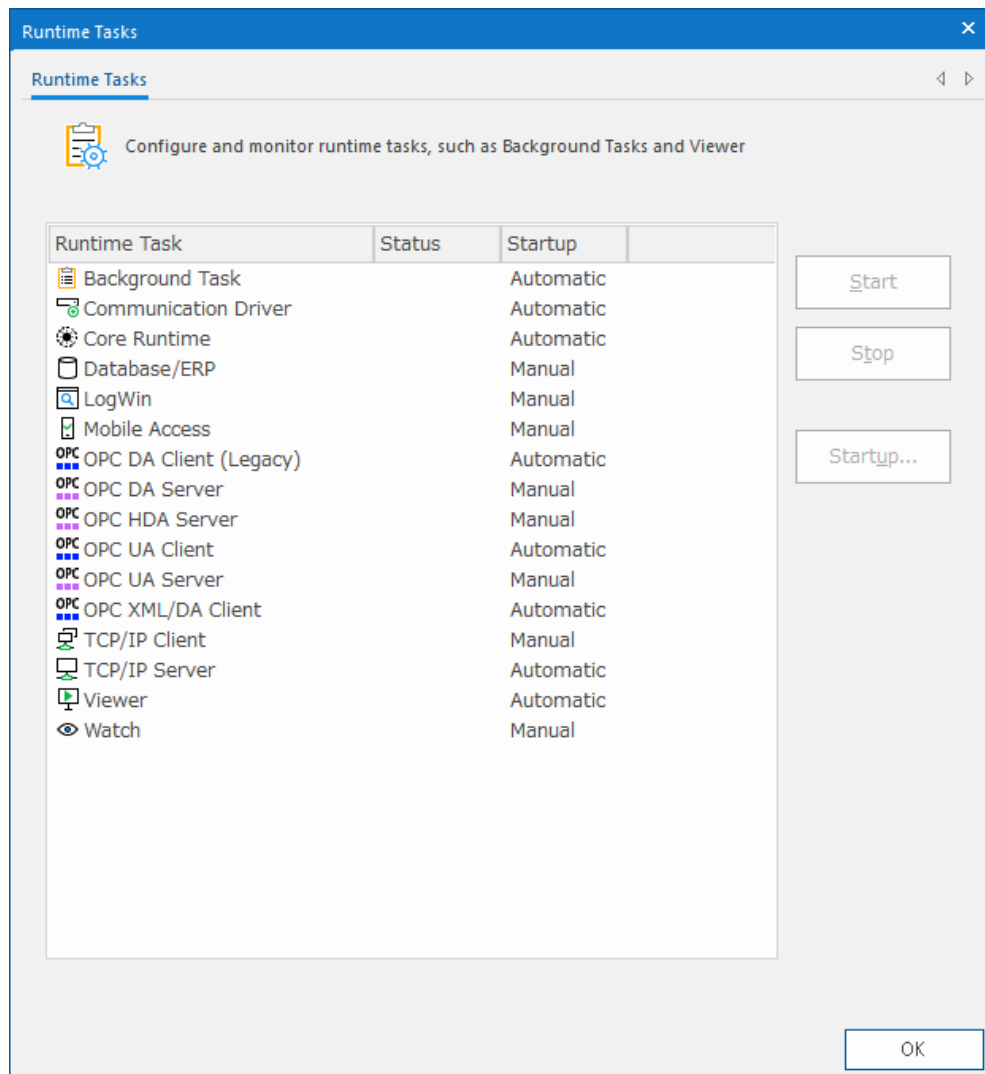
Checking the Communication Driver task

This section describes how to check the status of the Communication Driver task among the list of execution tasks.

The Communication Driver 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 yourself.

1. On the **Home** tab of the ribbon, in either the **Local Management** or the **Remote Management** group (depending on where the project server is running), click **Runtime Tasks**.

The *Runtime Tasks* dialog is displayed.



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

CHAPTER 4

Supported Registers

The ranges of supported register addresses are shown in the table below. Please note that the actual supported range of registers varies depending on the device that is used. Please check the actual range in the manual of your device.

Variable Data Type		bits	Bit Address	Word Address	32bits	Note
BOOL	Single Tag	1	<VARIABLENAME>		-	*1
	1D Array		<VARIABLENAME>[sx] ~ <VARIABLENAME>[ex]			
	2D Array		<VARIABLENAME>[sx,sy] ~ <VARIABLENAME>[ex,ey]			
	3D Array		<VARIABLENAME>[sx,sy, sz] ~ <VARIABLENAME>[ex,ey, ez]			
BYTE SINT USINT	Single Tag	8	<VARIABLENAME>.0 ~ <VARIABLENAME>.7	<VARIABLENAME>	L/H	*1
	1D Array		<VARIABLENAME>[sx].0 ~ <VARIABLENAME>[ex].7	<VARIABLENAME>[sx] ~ <VARIABLENAME>[ex]		
	2D Array		<VARIABLENAME>[sx,sy] .0 ~ <VARIABLENAME>[ex,ey] .7	<VARIABLENAME>[sx,sy] ~ <VARIABLENAME>[ex,ey]		
	3D Array		<VARIABLENAME>[sx,sy, sz].0 ~ <VARIABLENAME>[ex,ey, ez].7	<VARIABLENAME>[sx,sy,sz] ~ <VARIABLENAME>[ex,ey,e _z]		
INT UINT WORD	Single Tag	16	<VARIABLENAME>.00 ~ <VARIABLENAME>.15	<VARIABLENAME>	L/H	*1
	1D Array		<VARIABLENAME>[sx].00 ~ <VARIABLENAME>[ex].15	<VARIABLENAME>[sx] ~ <VARIABLENAME>[ex]		
	2D Array		<VARIABLENAME>[sx,sy] .00 ~ <VARIABLENAME>[ex,ey] .15	<VARIABLENAME>[sx,sy] ~ <VARIABLENAME>[ex,ey]		
	3D Array		<VARIABLENAME>[sx,sy, sz].00 ~ <VARIABLENAME>[ex,ey, ez].15	<VARIABLENAME>[sx,sy,sz] ~ <VARIABLENAME>[ex,ey,e _z]		

Variable Data Type		bits	Bit Address	Word Address	32bits	Note
REAL	Single Tag	32	-	<VARIABLENAME>	-	*1
	1D Array			<VARIABLENAME>[sx] ~ <VARIABLENAME>[ex]		
	2D Array			<VARIABLENAME>[sx,sy] ~ <VARIABLENAME>[ex,ey]		
	3D Array			<VARIABLENAME>[sx,sy,sz] ~ <VARIABLENAME>[ex,ey,ez]		
DINT UDINT DWORD	Single Tag	32	<VARIABLENAME>.00 ~ <VARIABLENAME>.31	<VARIABLENAME>	-	*1
	1D Array		<VARIABLENAME>[sx].00 ~ <VARIABLENAME>[ex].31	<VARIABLENAME>[sx] ~ <VARIABLENAME>[ex]		
	2D Array		<VARIABLENAME>[sx,sy] .00 ~ <VARIABLENAME>[ex,ey] .31	<VARIABLENAME>[sx,sy] ~ <VARIABLENAME>[ex,ey]		
	3D Array		<VARIABLENAME>[sx,sy, sz].00 ~ <VARIABLENAME>[ex,ey, ez].31	<VARIABLENAME>[sx,sy,sz] ~ <VARIABLENAME>[ex,ey,ez]		
STRING	Single Tag	(Number of char +1) x 8	-	<VARIABLENAME>	-	*1, *2, *3

*1: <VARIABLENAME>: Refers to the complete VARIABLENAME, including structure name, if it is a sub element of a structure. The complete variable name has a maximum length of 255 characters, including delimiters ('.') and address numbers (e.g. [1,2,3].12). The maximum length of the VARIABLENAME part only, or of a member name, is limited to 127 bytes.

e.g. BOOL type single Tag: "BOOLVAR"

e.g. BOOL array element: "BOOLARRAY[0012]"

e.g. INT type Tag: "INTVAR"

e.g. DINT type bit address: "DINTVAR.30"

e.g. REAL type 3D array: "REALARRAY[1,2,3]"

e.g. UINT member of a structure ("_sTimer"): "TIMERVAR.ET"

e.g. STRING type variable: "MYSTRINGVAR"

The following rules apply to the VARIABLENAME and Member Names.

- Maximum length: 127 Bytes
- Supported characters: Alphanumeric Characters (0~9, A~Z, a~z) and underscore (" _")
- The first character cannot be a number or the underscore character
- 2 consecutive underscore characters are not supported
- The following characters are not supported: ! " # \$ % & ' () = ~ ^ \ | ` @ { [+ ; * : }] < > , . ? / <Space>

*2: STRING type: When a Tag of type STRING is created or imported, it has an array parameter (1D only). STRING type Tags have a maximum array size of 1986.

*3: Does not correspond to an array.

Note: Reading or writing values from/to a single variable can take 2-3 milliseconds depending on the network and PLC environment.

Since the block read/write commands, which handle multiple variables in a single message, are not available on this driver, reading or writing values from/to 1000 variables can take 2-3 seconds.

When reading or writing many variables, using a one dimensional array is faster than individually setting multiple variables.

CHAPTER 5

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 OMNEP driver fails to communicate with the target device, then the database tags 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 the issue that occurred and how it might be resolved.

Error Code	Description	Possible Causes	Procedure To Solve
0 *1	OK	Communicating without error.	None required.
9	Error sending data	Connection error or problem sending data	Check the connections of the communication cable. Check the device data settings in the driver worksheet.
15	Timeout waiting for message to start	Timeout occurred	Check the timeout setting in the <i>Advanced settings</i> dialog and restart communication with the PLC
28	Read only device	Write to read only device	Check the device data settings in the driver worksheet.
29	Unsupported data type	The specified data type is invalid	Check the supported data types described in this document, and then correct the data type.
-35	Driver API not initialized	The driver library was not initialized by the driver.	Contact technical support.
-42	Invalid bit number	The bit number specified in the address is invalid. The limit for the bit number depends on the register type.	Check the addresses to see if there are bit numbers configured outside the valid range of the register.
-45	Invalid string size	The string is longer than 1024 characters.	Modify the addresses with the string data type to less than 1024 characters.
-54	Error sending data packet (TCP or UDP)	The TCP/UDP library was not able to send the packet.	Check the serial encapsulation settings.
-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.	Check the station IP address and port number. Confirm that the device is active and accessible. Try to ping the address.

*1: When the specified PLC variable name or bit number in the driver sheet is invalid (does not exist in the PLC), the value is displayed as "?", but the communication status remains 0.

Monitoring device communications

You can monitor the 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 later be downloaded.

Revision history

If you are unable to establish communication between Studio and the target device, try 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.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

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

Driver Version	Revision Date	Description of Changes
1.00	May, 2025	First driver revision