TXRX Communication Driver

Driver for Sending Data to or Receiving Data from
a Serial Port or TCP/IP Connection

Contents

Introduction

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

This document will help you to select, configure and execute the TXRX driver, and it is organized as follows:

- **Introduction**: This section, which provides an overview of the document.
- **General Information**: Identifies all of the hardware and software components required to implement communication between the Studio system and the target device.
- **Selecting the Driver**: Explains how to select the TXRX driver in the Studio system.
- **Configuring the Driver**: Explains how to configure the TXRX driver in the Studio system, including how to associate database tags with device registers.
- **Executing the Driver**: Explains how to execute the TXRX driver during application runtime.
- **Troubleshooting**: Lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.
- **Sample Application**: Explains how to use a sample application to test the TXRX driver configuration
- **Revision History**: Provides a log of all changes made to the driver and this documentation.

Notes:
- This document assumes that you have read the “Development Environment” chapter in Studio’s *Technical Reference Manual*.
- This document also assumes that you are familiar with the Microsoft Windows NT/2000/XP/Vista environment. If you are not familiar with Windows, then we suggest using the **Help** feature (available from the Windows desktop **Start** menu) as you work through this guide.

General Information

This chapter identifies all of the hardware and software components required to implement communication between the TXRX driver in Studio and remote devices.

The information is organized into the following sections:

- Device Specifications
- Network Specifications
- Driver Characteristics
- Conformance Testing

Device Specifications

You can use this driver to communicate with any device using a simple ASCII protocol. (The devices used for conformance testing are listed on the next page.)

Network Specifications

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

- **Device Communication Port**: RS232 Serial port or Ethernet port
- **Physical Protocol**: RS232/RS485 or TCP/IP
- **Logic Protocol**: ASCII or Binary
- **Device Runtime Software**: None
- **Specific PC Board**: None
- **Adapters/Converters**: Varies according to target device
- **Cable Wiring Scheme**: Varies according to target device

Driver Characteristics

The TXRX driver package consists of the following files, which are automatically installed in the **`\DRV`** subdirectory of Studio:

- **`TXRX.INI:`** Internal driver file. *You must not modify this file*.
- **`TXRX.MSG:`** Internal driver file containing error messages for each error code. *You must not modify this file*.
- **`TXRX.PDF:`** This document, which provides detailed information about the TXRX driver.
- **`TXRX.DLL:`** Compiled driver.

Note:
You must use Adobe Acrobat® Reader™ to view the **`TXRX.PDF`** document. You can install Acrobat Reader from the Studio installation CD, or you can download it from Adobe's Web site.

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

- Windows XP/Vista/7/8/10
- Windows CE 4.x, 5.x, 6.x, 7.x

For a description of the operating systems used to test driver conformance, see "Conformance Testing" below.

Conformance Testing

The following hardware/software was used for conformance testing:

- **Driver Configuration (a)**:
  - **PLC Program**: None
  - **Baud Rate**: 9600
  - **Protocol**: ASCII
  - **Data Bits**: 7
  - **Stop Bits**: 1
  - **Parity**: None
  - **COM Port**: 1
  - **TCP/IP Port**: 0
  - **Cable**: Serial Null-Modem

Driver Version	Studio Version	Operating System (development)	Operating System (runtime)	Equipment
1.21	8.0+SP1	Windows 8	Windows 8/CE	– Two PCs connected with a Serial cable – Two PCs connected with a TCP/IP connection

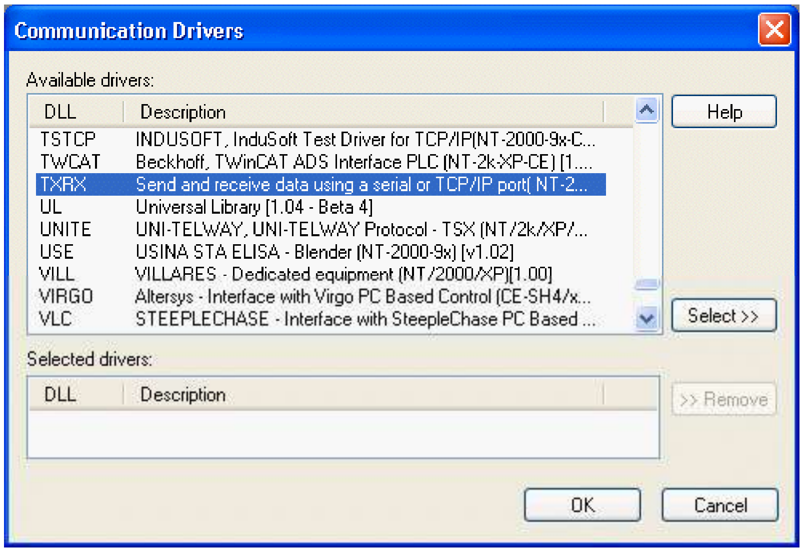
- **Driver Configuration (b)**:
  - **PLC Program**: None
  - **Baud Rate**: Not used
  - **Protocol**: ASCII
  - **Data Bits**: Not used
  - **Stop Bits**: Not used
  - **Parity**: Not used
  - **COM Port**: Not used
  - **TCP/IP Port**: 43981
  - **Cable**: Ethernet Cable

Driver Version	Studio Version	Operating System	Equipment/Software
1.21	8.0+SP1	Windows 8	– Two PCs connected with a Serial cable – Two PCs connected with a TCP/IP connection – GE FANUK, Schneider Electric M40 PLCs that talks MODBUS TCP/IP – Packet Sender

Selecting the Driver

When you install Studio, all of the communication drivers are automatically installed in the **\DRV** subdirectory but they remain dormant until manually selected for specific applications. To select the TXRX driver for your Studio application:

1. From the main menu bar, select **Insert** → **Driver** to open the *Communication Drivers* dialog.
2. Select the **TXRX** driver from the *Available Drivers* list, and then click the **Select** button.



DLL	Description
TSTCP	INDUSOFT, InduSoft Test Driver for TCP/IP(NT-2000-9x-C...
TWCAT	Beckhoff, TWinCAT ADS Interface PLC (NT-2k-XP-CE) [1....
TXRX	Send and receive data using a serial or TCP/IP port(NT-2...
UL	Universal Library [1.04 - Beta 4]
UNITE	UNI-TELWAY, UNI-TELWAY Protocol - TSX (NT/2k/XP/...
USE	USINA STA ELISA - Blender (NT-2000-9x) [v1.02]
VILL	VILLARES - Dedicated equipment (NT/2000/XP)[1.00]
VIRGO	Altersys - Interface with Virgo PC Based Control (CE-SH4/x...
VLC	STEEPLECHASE - Interface with SteepleChase PC Based ...

DLL	Description

Communication Drivers Dialog

3. When the **TXRX** driver is displayed in the **Selected Drivers** list, click the **OK** button to close the dialog. The driver is added to the *Drivers* folder, in the *Comm* tab of the Workspace.

Note:
It is not necessary to install any other software on your computer to enable communication between Studio and the target device. However, the TXRX driver is used only by Studio; you may need to install additional software to program the device.

Attention:
For safety reasons, you must take special precautions when installing any physical hardware. Please consult the manufacturer's documentation for specific instructions.

Configuring the Driver

Once you have selected the TXRX driver in Studio, you must properly configure it to communicate with your target device. First, you must set the driver's communication settings to match the parameters set on the device. Then, you must build driver worksheets to associate database tags in your Studio application with the appropriate addresses (registers) on the device.

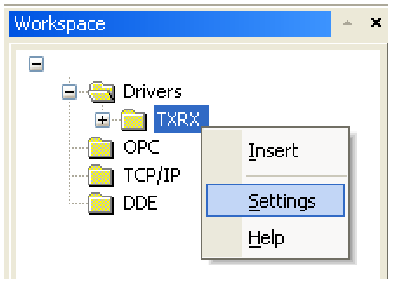
Configuring the Communication Settings

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

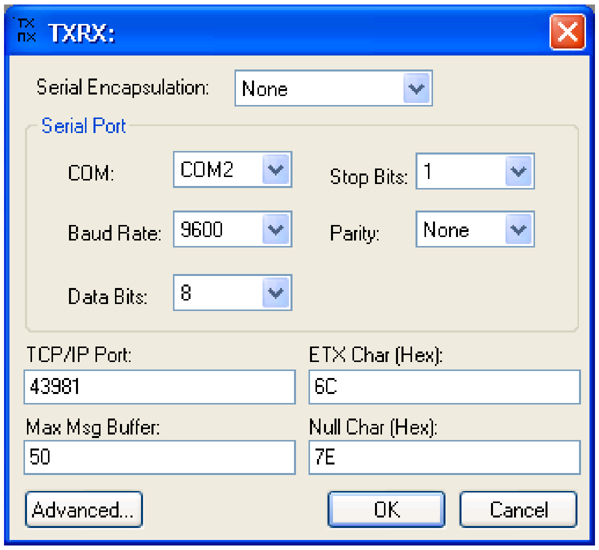
For the purposes of this document, only TXRX driver-specific settings and procedures will be discussed here. To configure the communication settings for the TXRX driver:

1. In the *Workspace* pane, select the *Comm* tab and then expand the *Drivers* folder. The TXRX driver is listed here as a subfolder.

2. Right-click on the *TXRX* subfolder and then select the **Settings** option from the pop-up menu. The *TXRX: Communication Parameters* dialog is displayed:



Select Settings from the Pop-Up Menu



Serial Encapsulation: None

Serial Port

COM: COM2
Stop Bits: 1
Baud Rate: 9600
Parity: None
Data Bits: 8

TCP/IP Port: 43981
ETX Char (Hex): 6C
Max Msg Buffer: 50
Null Char (Hex): 7E

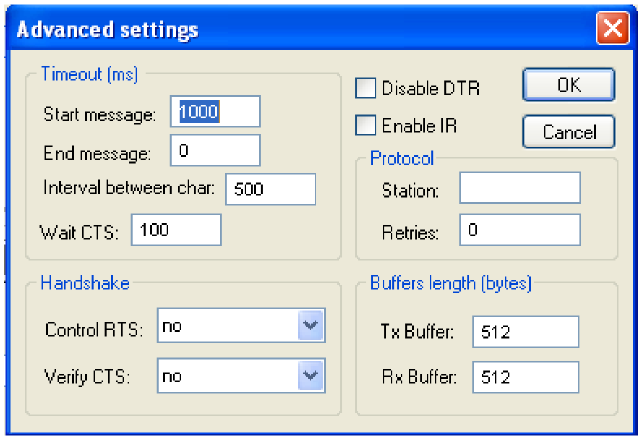
TXRX: Communication Parameters Dialog

3. Verify the *Serial Port* settings, and change them if necessary.

4. Configure the additional driver-specific settings, as described in the following table:

Setting	Default Value	Valid Values	Description
COM	`COM2`	`COM1` to `COM256`	Serial port of the PC used to communicate with the device.
Baud Rate	`9600`	`110` to `115200`	Communication rate of data.
Data Bits	`8`	`5` to `8`	Number of data bits used in the protocol. (ASCII is typically 7 bits; RTU is typically 8 bits.)
Stop Bits	`1`	`1` or `2`	Number of stop bits used in the protocol.
Parity	**`None`**	**`Even`**, **`Odd`**, **`None`**, **`Space`** or **`Mark`**	Parity of the protocol.
TCP/IP Port	0	Integer value	TCP/IP port number used to receive messages from other devices. ***Important***: If you want to use Serial communication, you must specify zero (`0`) for this parameter.
ETX Char (Hex)	0	`0` to **`FF`** **`N`** (None)	ETX character indicates the end of a message; Studio will put that character at the end of every Write command. You can specify multiple ETX characters using the following syntax: **`<1st ETX>`**, **`<2nd ETX>`**, **`<3rd ETX>`**... For example, you can specify `0D`, which is a carriage return in ASCII. Configure an **`N`** value if you do not want to use the ETX character.
Max Msg Buffer	**50**	Integer value	Number of messages that the driver will save in the internal buffer. If the buffer is full and new messages arrive, they will be ignored until the driver is finished processing at least one message.
Null Char		`01 to FF`	If the message that is arriving has the ASCII code zero (**0x00**), the driver will replace them by the ASCII code (in Hex format) specified in this field. To send zero (0x00) characters in a message you can set your string tag with the code specified (use function Asc2Str) in this field and it will be replaced by zero (0x00) in the send buffer. To send the value in this field in a message you will need to put it twice in the tag, i.e., if you specify 0x31 in this field (ASCII code for number 1) and you need to send a message with this code, your tag containing the message must duplicate the character (11 instead of 1). When a message is received and it contains the "Null Char" (value in this field) in it, the driver will duplicate the character so the user can distinguish between zero characters (0x00) and the actual "Null Char" specified in this field.

5. If you are using a Data Communication Equipment (DCE) converter (e.g., 232/485) between your PC and your target device, then you must also adjust the **Control RTS** (Request to Send) setting to account for the converter. In the *Communication Settings* dialog, click the **Advanced** button to open the *Advanced Settings* dialog:



Advanced settings

Timeout (ms)
Start message: 1000
End message: 0
Interval between char: 500
Wait CTS: 100

Disable DTR [ ]
Enable IR [ ]

Protocol
Station:
Retries: 0

Handshake
Control RTS: no
Verify CTS: no

Buffers length (bytes)
Tx Buffer: 512
Rx Buffer: 512

Advanced Settings Dialog

You do not need to change any other advanced settings at this time. You can consult the Studio *Technical Reference Manual* later for more information about configuring these settings.

6. Click **OK** to close the *Advanced Settings* dialog, and then click **OK** to close the *Communication Settings* dialog.

Configuring the Driver Worksheets

A selected driver includes one or more driver worksheets, which are used to associate database tags in Studio with operands on the target device. Each worksheet is triggered by specific application behavior, so that the tags / operands defined on that worksheet are scanned only when necessary – that is, only when the application is doing something that requires reading from or writing to those specific tags / operands. Doing this optimizes communication and improves system performance.

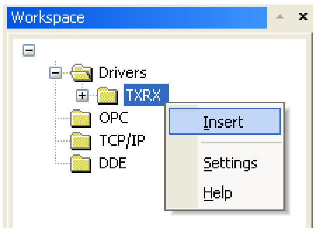
The configuration of these worksheets is described in detail in the “Communication” chapter of the Studio *Technical Reference Manual*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

Note:
We recommend configuring operands on the device in sequential blocks in order to maximize performance.

To insert a new driver worksheet:

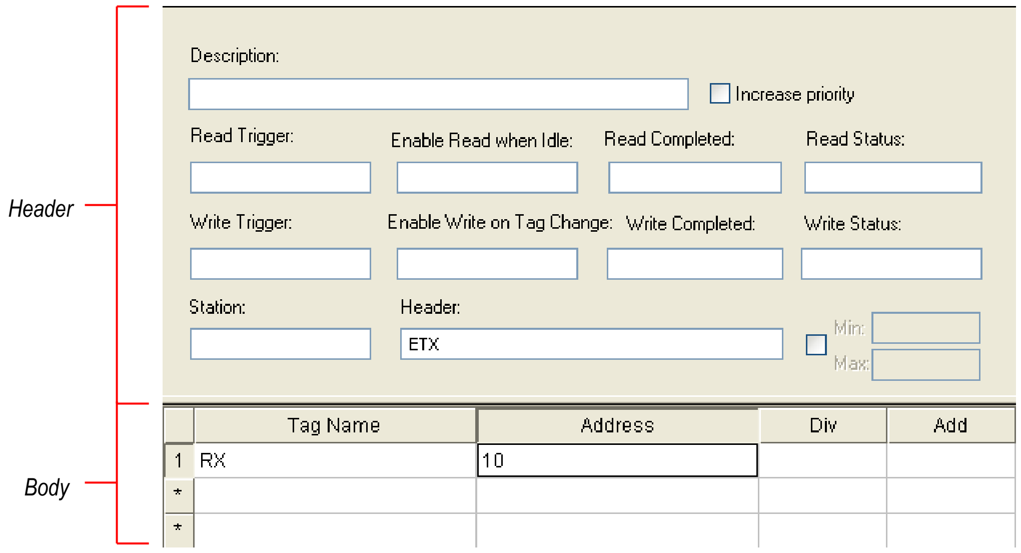
1. In the *Comm* tab, open the *Drivers* folder and locate the *TXRX* subfolder.

2. Right-click on the *TXRX* subfolder, and then select **Insert** from the pop-up menu:



Inserting a New Worksheet

A new TXRX driver worksheet is inserted into the *TXRX* subfolder, and the worksheet is opened for configuration:



	Tag Name	Address	Div	Add
1	RX	10		
*				
*				

TXRX Driver Worksheet

> **Note:**
> Worksheets are numbered in order of creation, so the first worksheet is **`TXRX001.drv`**.

Most of the fields on this worksheet are standard for all drivers; see the "Communication" chapter of the *Technical Reference Manual* for more information on configuring these fields. However, the **Station**, **Header**, and **Address** fields use syntax that is specific to the TXRX driver.

3. Configure the **Station** and **Header** fields as follows:

   - **Station** field (*not used for Serial communication*): Specify the IP Address of the device, using the following syntax:

     `<IP Address>[:Port Number]`

     Example — **`192.168.2.15:1111`**

     Where:

     – ***`<IP Address>`*** is the device's IP address on the TCP/IP network; and

     – ***`[Port Number]`*** is the TCP/IP port number to which you send messages. When configuring the station to receive value, the port number is taken from the *TXRX: Communication Parameters* dialog.

     There is no default value, and you can leave the field blank for Serial communication. 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.

   - **Header** field: Define the specific incoming and outgoing messages to be sent from the device.

     You can specify two types of headers for incoming messages:

     - **`RX`*`n`***: Writes incoming messages to a specified tag when ***`n`*** characters arrive.

- **ETX**: Writes incoming messages to a specified tag only when one of the **ETX** characters you configured for the driver arrives.

  RXTIMEOUT: Writes incoming messages to a specified tag only when there is no **ETX** char in the message and the *Interval Between Characters* arriving chars is elapsed (configured o*n Communication Parameters->Advanced*).

To specify a header for outgoing messages:

- **TX:*<ETX char (Hex value)>***: Sends outgoing messages to the device. Messages consist of a specified tag in ASCII format along with the specified **ETX** character.
- **HTX:*<ETX (Hex value)>***: Sends outgoing messages to the device. Message is written in hexadecimal value.

Note:
This header is a new functionality implemented in TxRx driver which allows this driver to write in hexadecimal mode. This mean that if a value like “12” is written to the tag, the driver will take this string that is in ASCII format and will convert this to a hexadecimal value. For example “12” that is ‘1’ and ‘2’ in ASCII, so the driver will write 0x12.
The maximum hexadecimal value accepted in this case is 0xFF (255 in decimal), values beyond this limit will be divided. Example: if a value like “1234” is passed to the driver, it will write 0x12 0x34.

Other examples:

Value written in the Tag	Value that driver will write
“1”	0x1
“12”	0x12
“123”	0x12 0x3
“12 34”	0x12 0x34
“12 34 5”	0x12 0x34 0x05
“ 5 6 7 8 ”	0x05 0x06 0x07 0x08
“12345”	0x12 0x34 0x05

For example:

- **TX** sends messages only. If you do not specify an **ETX** character in the **Header**, Studio automatically uses the **ETX** character from the **Settings** field for Write commands. (For example: **TX:0D**)
- **TX:0A** sends messages only. Studio uses the **0A** character (as the **ETX** character) at the end of the outgoing message.
- **RX10** receives incoming messages of ten (10) characters.
- **ETX** receives incoming messages ending with one of the **ETX** characters you specified in the *Communication Parameters* dialog.
- **RXTIMEOUT** receives incoming messages with the data in the internal buffer if the buffer does not contain any ETX and no other header was found to handle the message. This condition is

checked after the interval between characters configured in *Communication Parameters->Advanced* dialog

- □ **`HTX`** write for example the value 10 in the tag, the driver will send 0x10 to the device.

You can type **`{Tag}`** into the **Header** field, but you must be certain that the Tag's value is correct and that you are using the correct syntax, or you will get an invalid Header error.

Note: Studio transfers incoming messages to tags in the tag fields as strings.

Note:
For TxRx driver, the Header's field should not be changed after starting the driver execution.

4. For each table row (i.e., each tag/register association), configure the **Address** field using the following syntax:
   - □ For Write commands, the address must specify the number of characters that can be sent for each line of the worksheet. The maximum value per line is 80 characters.
   - □ For incoming messages or responses, use the following syntax
     `L:`*`<Message Length>`*`, S:`*`<Separator Char>`* or ***`<Any Integer Value>`***
     (For example: **`L:10`**)
   - □ In case of hexadecimal format output, use the following syntax
     `HL:`***`<Message Length>`***`, HS:`***`<Separator Char>`***
     (For example: **`HL:2  HS:32`**)
   - □ For showing all the receive values in hexadecimal format :
     `H`
     (For example: `H`)

Note:
In hexadecimal format the maximum numbers of values that can be displayed are 341 hexadecimal values. Exceeded the limit of size, it reports a "warning", stating that data loss is occurring.

Where:

- ***`<Message Length>`*** is the number of characters (starting with the last character defined in the Driver Worksheet) the address tag will receive.
- ***`<Separator Char>`*** is an ASCII character (hexadecimal) used to end and separate messages; for example, **`S:41`**.
- ***`<Any Integer Value>`*** is the number of characters (starting with the last character defined in the Driver Worksheet) the address tag will receive.

Header Field	Address Field
ETX	**L:10**
ETX	**S:20**
ETX	**S:20**
ETX	**L:10**

Note:

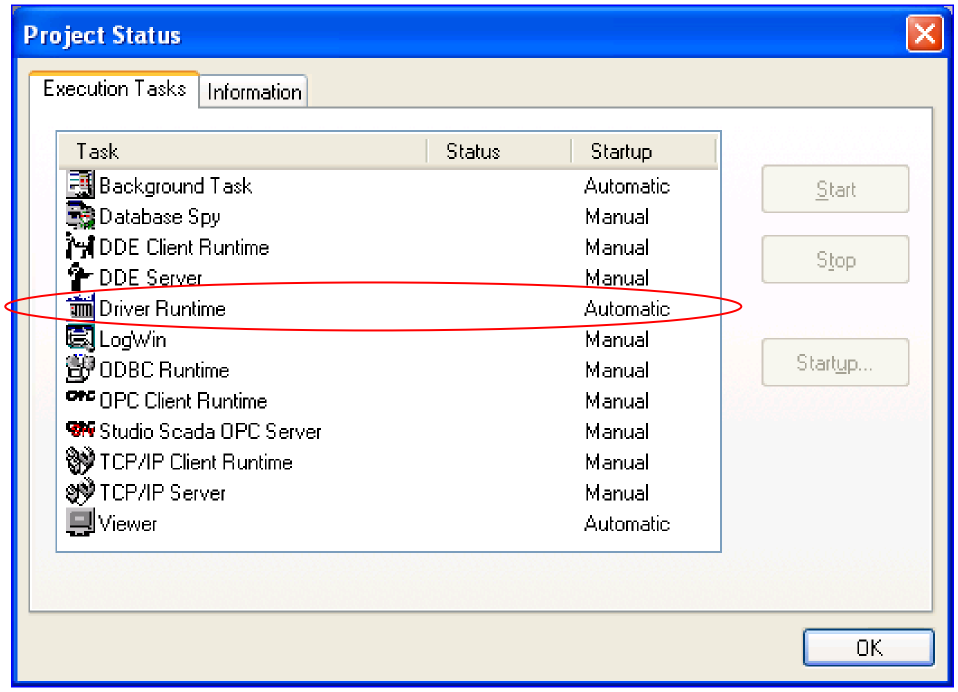
For Header types ETX, RXn and RXTIMEOUT, the IP address of the Source device can be defined on the Station field of the driver worksheet. The TCP/IP Port is not passed on the Station field in these cases, since the Port used for incoming messages is the one configured under Driver Settings. In this case, only values received from the specific source IP will be assumed. The IP address of the Source device is required in case the project has at least two (or more) driver worksheets with the same Header type (e.g. ETX, RXn or RXTIMEOUT). In case the project has only one single driver worksheet for one of the aforementioned headers, the station field is ignored and the tag of the respective worksheet receives incoming values from any Source device arriving on the TCP/IP Port configured on the Driver Settings.

Executing the Driver

By default, Studio will automatically execute your selected communication driver(s) during application runtime. However, you may verify your application's runtime execution settings by checking the *Project Status* dialog.

To verify that the the communication driver(s) will execute correctly:

1. From the main menu bar, select **Project** → **Status**. The *Project Status* dialog displays:



Task	Status	Startup
Background Task		Automatic
Database Spy		Manual
DDE Client Runtime		Manual
DDE Server		Manual
Driver Runtime		Automatic
LogWin		Manual
ODBC Runtime		Manual
OPC Client Runtime		Manual
Studio Scada OPC Server		Manual
TCP/IP Client Runtime		Manual
TCP/IP Server		Manual
Viewer		Automatic

Project Status Dialog

2. Verify that the *Driver Runtime* task is set to **Automatic**.

   - If the setting is correct, then proceed to step 3 below.
   - If the **Driver Runtime** task is set to **Manual**, then select the task and click the **Startup** button to toggle the task's *Startup* mode to **Automatic**.

3. Click **OK** to close the *Project Status* dialog.

4. Start the application to run the driver.

Troubleshooting

If the TXRX driver fails to communicate with the target device, then the database tag(s) that you configured for the **Read Status** or **Write Status** fields of the Standard Driver Sheet will receive an error code. Use this error code and the following table to identify what kind of failure occurred.

Error Code	Description	Possible Causes	Procedure to Solve
0	OK	Communicating without problems	Not required
2	Failure to allocate memory	Not enough memory to allocate the driver buffer	Increase the system memory.
10	Invalid Header	Invalid Header provided or tag in the **Header** field has an invalid configuration	Type a valid Header or a valid tag value in the **Header** field. Consult the Studio *Technical Reference Manual* for a list of valid headers.
16	Invalid command	Invalid command specified in the Driver Worksheet	Disable all **Read** fields in the Driver Worksheet.
17	Invalid ETX	Invalid **ETX** value specified	Type a valid **ETX** value (between 01 and FF).
18	Error connecting	Invalid IP address or TCP/IP port specified	Type a valid IP Address (TCP/IP) in the **Station** field or for the tag value.
19	Generic TCP/IP Error	TCP/IP layer seems to be malfunctioning	- Check if your network card is working properly - Check if the TCP/IP protocol is properly installed on your network settings
20	Invalid value, please specify a value between 1 and 50	Configuration the driver Settings *Max Msg Buffer* parameter is outside the range between 1 and 50 messages	Please enter a value between 1 and 50
21	Invalid Null Char	Configuration on the driver Settings *Null Char* has a invalid value, usually a non-Hex format value	Please enter a valid char code in Hex format

⇨ **Tip:**

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 **Protocol Analizer,** right-click in the *Output* window and select the desired options from the pop-up menu.

You can also use the *LogWin* module (**Tools** → **LogWin**) to establish an event log on a remote unit that runs Windows CE. 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.

If you must contact us for technical support, please have the following information available:

- **Operating System** (type and version): To find this information, select **Tools** → **System Information.**
- **Project Information**: To find this information, select **Project** → **Status.**
- **Driver Version** and **Communication Log**: Displays in the Studio *Output* window when the driver is running.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

Sample Application

A sample application that employs the TXRX driver is provided on the Studio installation CD. We strongly recommend that you use this sample application to test the driver *before* you develop your own applications, for the following reasons:

- To better understand the information and instructions provided in this document;
- To verify that your driver configuration is working satisfactorily with the target device; and
- To ensure that the all of hardware used in the test (i.e. the device, adapter, cable, and PC) is functioning safely and correctly.

Note:
The following instructions assume that you are familiar with developing project applications in Studio. If you are not, then please review the relevant chapters of the Studio *Technical Reference Manual* before proceeding.

To use the sample application:

1. Configure the device's communication settings according to the manufacturer's documentation.
2. Run Studio.
3. From the main menu bar, select **File → Open Project**.
4. Insert the Studio installation CD and browse it to find the sample application. It should be located in the directory **`\COMMUNICATION EXAMPLES\TXRX`**.
5. Select and open the sample application.
6. Configure and test the driver, as described in the rest of this document.

When you have thoroughly tested the driver with your target device, you may proceed with developing your own Studio application projects.

Tip:
You can use the sample application screen as the maintenance screen for your own applications.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of Changes
A	1.00	Lourenço Teodoro	9 Mar 2001	Initial version
B	1.01	Lourenço Teodoro	27 Apr 2001	Improved unsolicited messages treatment
C	1.02	Lourenço Teodoro	19 Nov 2001	▪ Fixed the lock-up bug ▪ Implemented the receiving feature with more than one line in the address
D	1.03	Robert Vigiani Jr.	5 Dec 2001	▪ Included TCP/IP communication ▪ Included configuration of ETX char in the send messages ▪ Included message treatment by length or by separator char
E	1.04	Lourenço Teodoro	3 May 2002	The parameter Interval Between RX char is used to reset the RX buffer. The previous version had a fixed time of 500ms.
F	1.05	Lourenço Teodoro	9 Oct 2002	Fixed bug when using Interval Between RX char with TCP/IP
G	1.06	Bryan Morgan	14 Feb 2003	Fixed bug preventing the driver from using the ETX char when sending data
H	1.07	Lourenço Teodoro	01 Dec 2003	Fixed send problem when the driver was compiled for UNICODE. This problem could happen under Windows CE.
I	1.08	Bruno A. Crepaldi	12 Nov 2004	Fixed the problem of memory leak
J	1.09	Leandro Coeli	06 Sep 2005	Fixed the ETX problem
K	1.10	Eric Vigiani	04 Oct 2006	Modified the driver to accept multi-connections
L	1.10	Michael D. Hayden	08 Dec 2006	Edited for language and usability.
M	1.11	Jonathan C. Romanus	14 Feb 2007	▪ Fixed problem when receiving 8X characters ▪ Changed the driver to be only UNICODE ▪ Fixed duplicated first character when time between received messages was less than one second.
N	1.11	Rafael R Fernandes	15 Mar 2007	▪ Driver modified to accept binary messages.
O	1.12	Plínio M. Santana	23 Nov 2007	▪ Included exclusive ETX option on the Communication Parameters. ▪ Included Ignored Messages Amount option on the Communication Parameters.
P	1.12	Plinio M. Santana	23 Jan 2008	▪ Fixed RX Buffer to don't repeat the ETX char. ▪ Modification to synchronize operations.
Q	1.14	Lourenco Teodoro	5 Jan 2009	▪ Fixed problem with ETX higher than 8F ▪ Modified Communication Parameters window ▪ Added Max Msg Buffer and Null Char parameters ▪ Added RXTIMEOUT Header
R	1.15	Paulo R. Balbino	9 Jul 2009	▪ Added hexadecimal support ▪ Fixed problem with RXn header ▪ Increased message size for up to 1024 characters ▪ Fixed problem when using RXTIMEOUT header and there are Null (0x00) characters in the message

S	1.15	Andre Bastos	10 Dec 2009	▪ Documentation revised only. No modifications on the driver
T	1.16	Eric Vigiani	10 Dec 2009	▪ Fixed issue with ETX and RX Headers
U	1.17	Paulo Balbino	26 Jul 2010	▪ Fixed issue with adding an extra character to the TX message
V	1.18	Lourenço Teodoro	25 Mar 2011	▪ Fixed issue to receive data when connecting as a client under Windows CE.
W	1.19	Ricardo Marroni / Vivek Abraham Anushree Phanse	18 Oct 2016	▪ Implemented support for source IP address for Headers ETX, RXn and RXTIMEOUT.
X	1.20	Anushree Phanse	03 Nov 2016	▪ Driver project changed to fix compilation issues, version incremented.
Y	1.21	Anushree Phanse	05 Apr 2017	▪ Fixed issue of driver not being able to to reconnect with the client when using serial encapsulation with TCP/IP as a Server by fixing Unicomm. ▪ Removed invalid diagnostic log messages seen on starting the driver.