

DAKOT Communication Driver

Driver for TCP/IP Communication
with RTU devices from Dakota Fluid Power

Contents

INTRODUCTION2

GENERAL INFORMATION.....3

 DEVICE SPECIFICATIONS.....3

 NETWORK SPECIFICATIONS.....3

 DRIVER CHARACTERISTICS3

 CONFORMANCE TESTING4

SELECTING THE DRIVER5

CONFIGURING THE DRIVER6

 CONFIGURING THE COMMUNICATION SETTINGS6

 CONFIGURING THE DRIVER WORKSHEETS8

EXECUTING THE DRIVER 13

TROUBLESHOOTING 14

REVISION HISTORY..... 15

Introduction

The DAKOT driver enables communication between the Studio system and RTU systems from Dakota Fluid Power using TCP over IP, according to the specifications discussed in this document.

This document will help you to select, configure and execute the DAKOT 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 DAKOT driver in the Studio system.
- **Configuring the Driver:** Explains how to configure the DAKOT driver in the Studio system, including how to associate database tags with device registers.
- **Executing the Driver:** Explains how to execute the DAKOT driver during application runtime.
- **Troubleshooting:** Lists the most common errors for this driver, their probable causes, and basic procedures to resolve them.
- **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 environment.

General Information

This chapter identifies all of the hardware and software components required to implement communication between the DAKOT driver in Studio and a target RTU system using TCP over IP.

The information is organized into the following sections:

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

Device Specifications

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

- **Manufacturer:** Dakota Fluid Power
- **Compatible Equipment:**
 - RTU systems using TCP over IP
- **Device Runtime Software:** None

For a description of the device(s) used to test driver conformance, see “Conformance Testing” on the next page.

Network Specifications

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

- **Device Communication Port:** Ethernet Port or modem converting RTU data to TCP over IP
- **Physical Protocol:** Ethernet
- **Logic Protocol:** TCP/IP
- **Specific PC Board:** Any TCP/IP Adapter (Ethernet board)

Driver Characteristics

The DAKOT driver package consists of the following files, which are automatically installed in the `/DRV` subdirectory of Studio:

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

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

- Windows NT (e.g. Windows XP/7/8/2003/2008/2012)

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

The DAKOT driver supports the following registers:

Register Type	Length	Write	Read	Bit	Integer	String
Boolean	1 Bit	•	•	•	•	–
Byte	1 Byte	•	•	•	•	–
Word	1 Word	•	•	•	•	–
Double Word	2 Words	•	•	•	•	–
String	1 Byte/Char	•	–	–	–	•

Conformance Testing

The following hardware/software has been used for conformance testing:

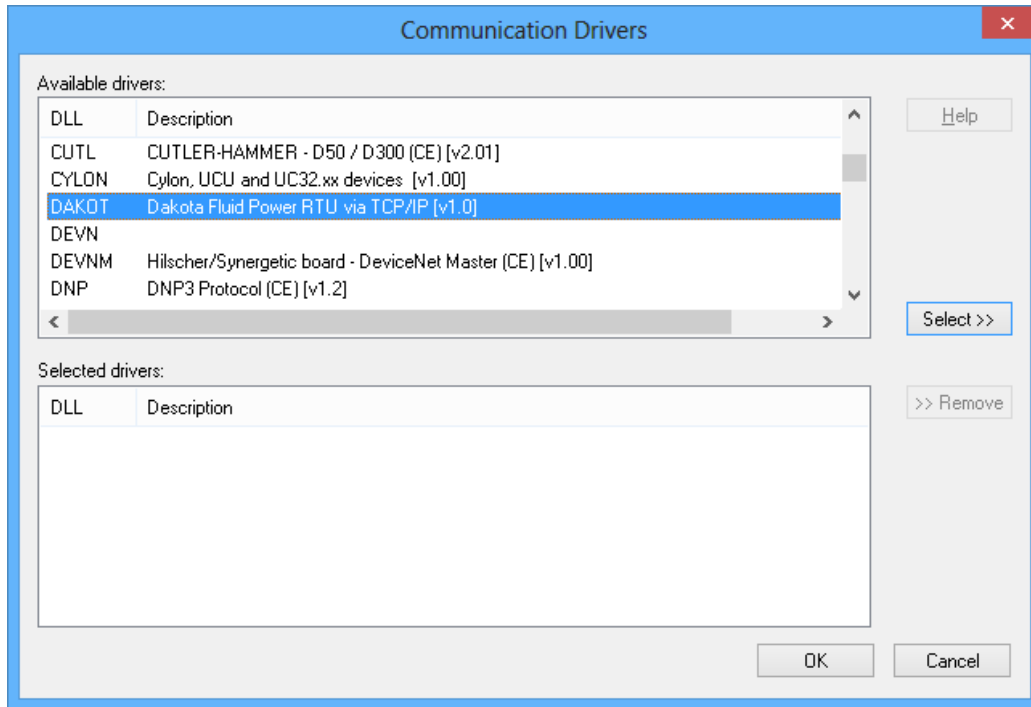
- **Equipment:** Dakota Fluid Power has setup a modem sending RTU data over TCP/IP on port 4036 to the InduSoft Web Studio workstation.
- **Driver Configuration:**
 - **Port:** 4036
 - **Cable:** Ethernet Cable

Driver Version	Studio Version	Operating System (development)	Operating System (runtime)	Equipment
1.2	8.0	Windows 7 + SP1 x64	▪ Win7 x64	Dakota Fluid Power RTU over TCP/IP modem

Selecting the Driver

When you install Studio, all of the communication drivers are automatically installed in the `\DRV` subdirectory. Users must manually select the driver for specific applications in order to have it applied to the project. To select the DAKOT driver for your Studio application:

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



Communication Drivers Dialog

3. When the **DAKOT** 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 your target device. However, this communication can only be used by the Studio application; it cannot be used to download any control logic to the device. For more information, please consult the documentation provided by the device manufacturer.

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

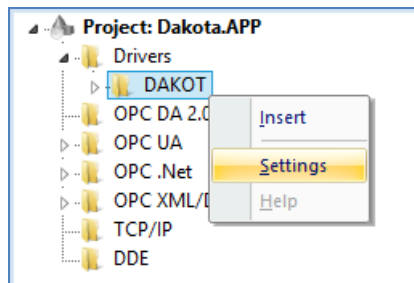
Once you have selected the DAKOT driver in Studio, you must properly configure it to communicate with your target device. First, you must set the driver’s communication settings to match the parameters set on the device. Then, you must build driver worksheets to associate database tags in your Studio 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*, and the same general procedures are used for all drivers. Please review those procedures before continuing.

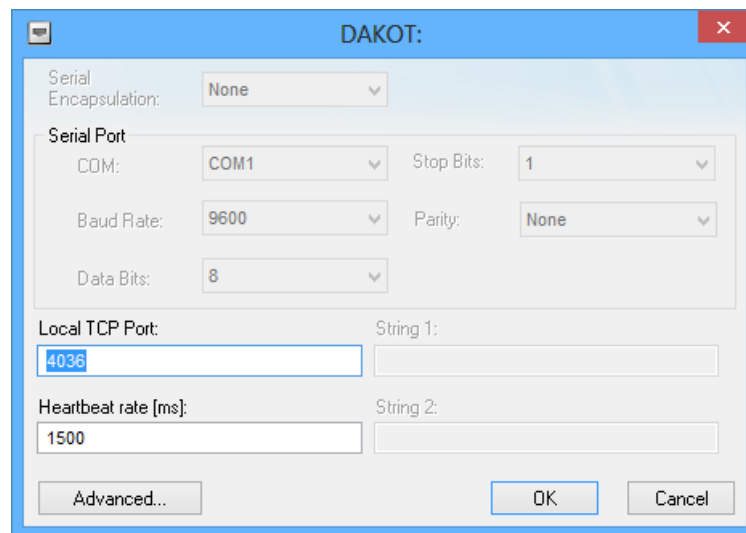
For the purposes of this document, only DAKOT driver-specific settings and procedures will be hereby described. To configure the communication settings for the DAKOT driver:

1. In the *Workspace* pane, select the *Comm* tab and then expand the *Drivers* folder. The DAKOT driver is listed as a subfolder.
2. Right-click on the *DAKOT* subfolder and then select the **Settings** option from the pop-up menu:



Select Settings from the Pop-Up Menu

The *DAKOT Communication Settings* dialog is displayed:



DAKOT Communication Settings Dialog

- In the *Communication Settings* dialog, configure the driver settings to enable communication with your target device. To ensure error-free communication, the driver settings must *exactly match* the corresponding settings on the device. Please consult the manufacturer’s documentation for instructions how to configure the device and for complete descriptions of the settings.

Depending on your circumstances, you may need to configure the driver *before* you have configured your target device. If this is the case, then take note of the driver settings and have them ready when you later configure the device.

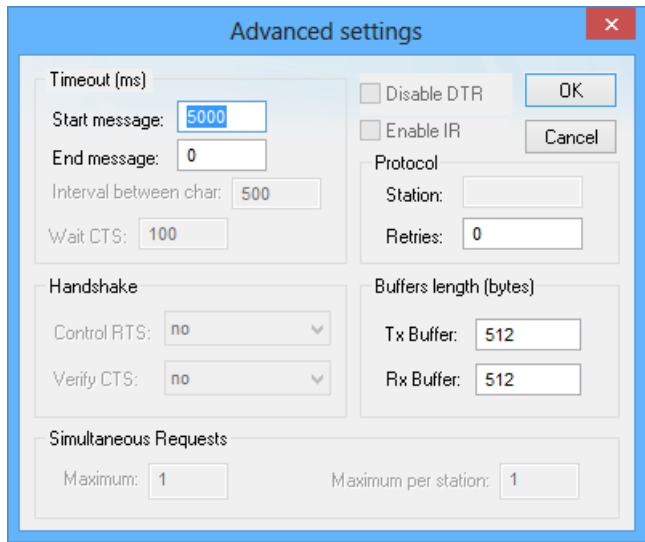
Attention:
 For safety reasons, you **must** take special precautions when connecting and configuring new equipment. Please consult the manufacturer’s documentation for specific instructions.

The communication settings and their possible values are described in the following table:

Parameters	Default Values	Valid Values	Description
Local TCP Port	4036	1 to 65535	The local TCP port to which the driver will listen in order to establish communication with the RTU or RTU-Modem.
Heartbeat rate [ms]	0	0 to 2147483647	Specify the time rate in milliseconds that the driver sends the Heartbeat information to the RTU. If the value is set to zero or negative, Heartbeat will be disabled on the driver. By default, Heartbeat is disabled.

Note:
 The device must be configured with *exactly the same* parameters that you configured in the *DAKOT Communication Parameters* dialog.

- In the *Communication Settings* dialog, click the **Advanced** button to open the *Advanced Settings* dialog:



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.

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

Configuring the Driver Worksheets

A Studio driver may include a Main Driver Sheet and one or more Standard Driver Worksheets. For the DAKOT driver only Standard Driver Sheets are enabled. Standard Driver Worksheets can be inserted to define tag/register associations that are triggered by specific application behaviors as well as for messages which does not require any trigger (unsolicited messages).

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

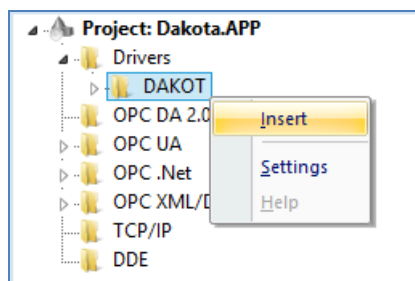
For the purposes of this document, only DAKOT driver-specific parameters and procedures are hereby described.

STANDARD DRIVER WORKSHEET

When you select the DAKOT driver and add it to your application, it has no Driver Sheet by default (see previous section). However, you may insert Standard Driver Worksheets to define tag/register associations that are triggered by specific application behaviors as well as for the messages which does not require any trigger such as unsolicited messages.

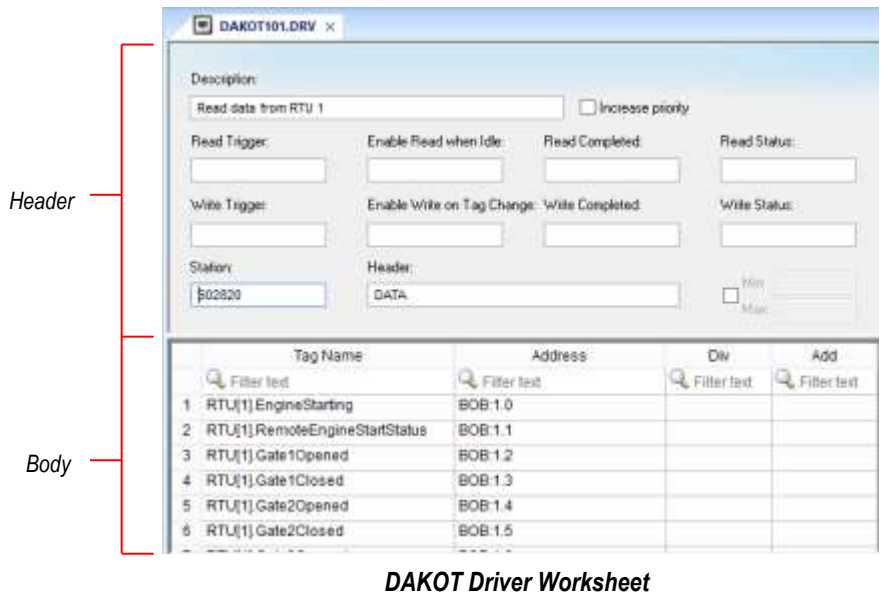
To insert a new Standard Driver Worksheet:

1. In the *Comm* tab, open the *Drivers* folder and locate the *DAKOT* subfolder.
2. Right-click on the *DAKOT* subfolder, and then select **Insert** from the pop-up menu:



Inserting a New Worksheet

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



DAKOT Driver Worksheet

Note:
 Worksheets are numbered in order of creation, so the first worksheet is **DAKOT001.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 syntaxes that are specific to the DAKOT driver.

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

- **Station** field: Specify the RTU unique ID of the device, using the following syntax:

<RTUID>

Example — 602820

Where:

- **<RTUID>** is the RTU unique identification number (from 1 to 4294967295).

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.

Note:
 For the Header type RTUINFO, the value zero (“0”) on the station field is also acceptable. In this case, any information data received by the driver, i.e. from every RTU connected, will be assigned to the tags listed on the driver worksheet body.

- **Header** field: Specify the type of information to be read or written from/to the RTU.

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

<Type>[:<CodeNumber>]

Where:

- **<Type>** is the information type (DATA, RTUINFO or CMD)
- **<CodeNumber>** is the command code used by the **Type** CMD only to send commands to the RTU (e.g. 201, 202, 203).

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

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

Examples:

Header	Description
DATA	Read specific RTU unsolicited messages.
RTUINFO	Read RTU information data.
CMD:202	Write command with the code number 202 to the RTU.

 **Note:**

For the worksheets with header type DATA and RTUINFO the driver will read the unsolicited messages automatically and no trigger is required. For the header type CMD, the Driver will execute a writing command only when the Write Trigger command is executed.

4. For each table row (i.e., each tag/register association), configure the **Address** field using the following syntax...

For header type DATA use the following syntax:

<MsgType>[<Format>]:<Address>[.<Bit>]

Examples — **BOB:1.2, WD:10, WDW:5, DWD:3, DWB:2**

For header type RTUINFO use the following syntax:

<Info>

Example — **RTUID, IP, AVGSCAN**


For header type CMD use the following syntax:

[<Format>][.<Bit>]

Example — **W, B, B.2, D, S, 0**

Where:

- **<MsgType>**: Type of message being received from RTU (e.g. **BY, BO, WD, CW, DW, SE**).
- **[<Format>]** (optional): Optional parameter that defines the portion of the received message that will be written to the tag (e.g. **B, W, D**). If not defined, the driver uses **Byte**.
- **<Address>**: Number of the initial byte from the message for the value that will be written to the tag..
- **[.<Bit>]** (optional): Optional parameter with the bit position that must be extracted from the message (0..31).
- **<Info>**: Type of information that is required from the message.

 **Note:**
 For commands (header type **CMD**) which do not require any supporting information value, you must enter value zero (“0”) at the **Address** field of the first row on the driver worksheet body (the **Tag** field can be blank). Otherwise the driver will not send the command message to the RTU.

Examples of appropriate **Header** and **Address**:

Register Address and Message Type on the RTU	Header	Address
Read Byte 1 of Message Type Byte	DATA	BY : 1
Read Bit 2 of Byte 1 of Message Type Byte	DATA	BY : 1 . 2
Read Byte 20 of Message Type Byte	DATA	BYB : 20
Read Byte 5 of Message Type Word	DATA	WDB : 5
Read Word 3 of Message Type Word	DATA	WDW : 3
Read Bit 1 of Byte 4 of Message Type Double Word	DATA	DWB : 4 . 1
Read Double Word 1 of Message Type Double Word	DATA	DWD : 1
Read Bit 23 of Double Word 1 of Message Type Double Word	DATA	DWD : 1 . 23
Read Bit 0 of Byte 1 of Message Type BOOL	DATA	BOB : 1 . 0
Read Bit 7 of Byte 1 of Message Type BOOL	DATA	BOB : 1 . 7
Read RTU unique ID	RTUINFO	RTUID
Read RTU IP Address	RTUINFO	IP
Read Local RTU Average Scan time	RTUINFO	AVGSCAN
Read Last RTU Cycle Scan Time	RTUINFO	CYCLESKAN
Read Security Scan Time OK	RTUINFO	SCANSTATUS
Read Local RTU Library Major Version	RTUINFO	VERSIONMAJOR
Read Local RTU Library Minor Version	RTUINFO	VERSIONMINOR
Read Local High Speed Coms	RTUINFO	HIGHSPEEDCOMS
Read Machine Doing Work	RTUINFO	MACHINE_STATUS

Register Address and Message Type on the RTU	Header	Address
Write Command code 200 with supporting information value as Byte to the RTU	CMD : 200	B
Write Command code 201 with supporting information value as Byte to the RTU	CMD : 201	B
Write Command code 202 with supporting information value as Word to the RTU	CMD : 202	W
Write Command code 203 without supporting information value to the RTU	CMD : 203	0
Write Command code 204 with supporting information value as Word to the RTU	CMD : 204	W
Write Command code 206 without supporting information value to the RTU	CMD : 206	0
Write Command code 212 without supporting information value to the RTU	CMD : 212	0
Write Command code 251 with supporting information value as String (ascii code of each character will be sent in bytes)	CMD : 251	S

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

 **Note:**

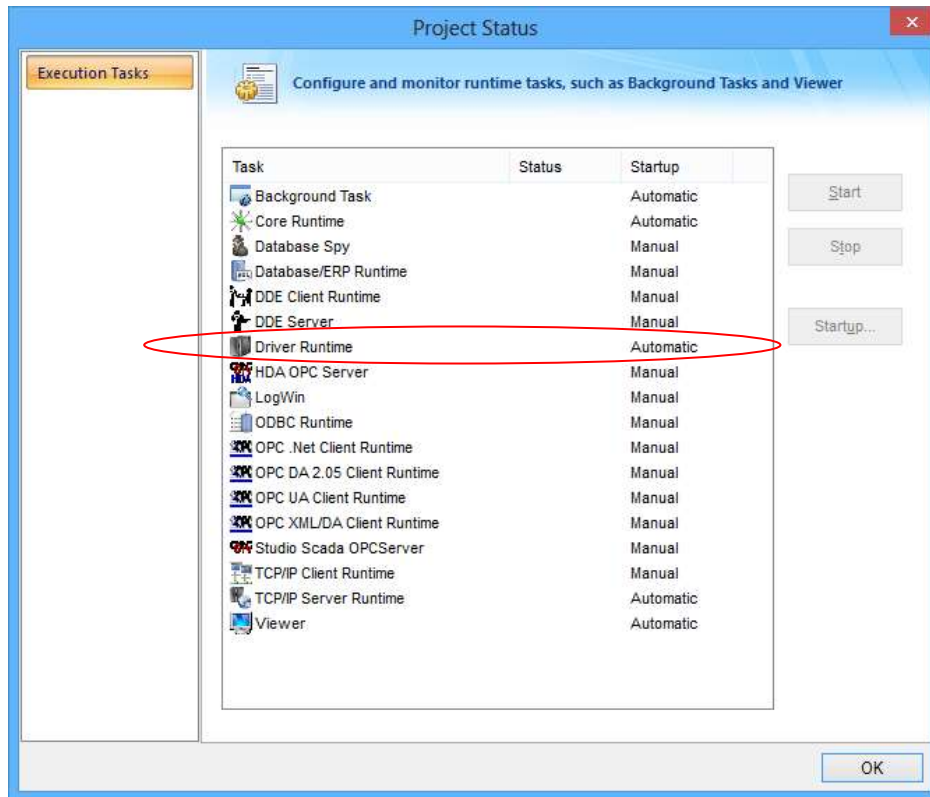
For the command code 203 (Engine Start Command), the driver sends the supporting information value (“START”) automatically. Therefore you should not enter any supporting information value for this specific command. For all other commands requiring supporting information, the value must be given as described on the table above.

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 ribbon, select **Home tab** → **Tasks** under *Local Management*. The *Project Status* dialog pops up:



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 DAKOT 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 Driver Sheet will receive an error code.

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

You can also use the *Remote LogWin* module to establish an event log on a remote unit that runs on a remote runtime, including Windows CE and Embedded

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 information generated by the command **Support Information** in Studio's **Help** menu ready.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of changes
A	1.0	Ricardo Marroni	April 8, 2014	Document creation
B	1.1	Eduardo Castro	September 9, 2014	Fixed issues to get station field changes.
C	1.2	Anushree Phanse	October 22, 2015	Fixed issues on the RTU table and CSV file Fixed an issue on Write for Groups and on the string conversion