PNOZ Communication Driver

Driver for Serial Communication
with PNOZmulti Safety System Device

Contents

Introduction

The PNOZ driver enables communication between the Studio system and devices using the PNOZmulti safety system by serial communication, according to the specifications discussed in this document.

This document was designed to help you install, configure, and execute the PNOZ driver to enable communication with these devices. The information in this document is organized as follows:

- **Introduction**: Provides an overview of the PNOZ driver documentation.
- **General Information**: Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the PNOZ driver.
- **Installing the Driver**: Explains how to install the PNOZ driver.
- **Configuring the Driver**: Explains how to configure the PNOZ driver.
- **Executing the Driver**: Explains how to execute the driver to verify that you installed and configured the driver correctly.
- **Troubleshooting**: Lists the most common error codes for this protocol and explains how to fix these errors.
- **Sample Application**: Explains how to use a sample application to test the PNOZ driver configuration.
- **Revision History**: Provides a log of all modifications made to the driver and the documentation.

> **Notes:**
> - This document assumes that you have read the “Development Environment” chapter in the Studio *Technical Reference Manual*.
> - This document also assumes that you are familiar with the Windows NT/2000/XP environment. If you are unfamiliar with Windows NT/2000/XP, we suggest using the **Help** feature (available from the Windows desktop **Start** menu) as you work through this guide.

General Information

This chapter explains how to identify all the hardware and software components used to implement communication between the Studio PNOZ driver and the PNOZmulti safety system devices.

The information is organized into the following sections:

- Device Characteristics
- Link Characteristics
- Driver Characteristics

Device Characteristics

To establish communication, you must use devices with the following specifications:

- **Manufacturer**: PILZ
- **Compatible Equipment**: PNOZmulti Safety System
- **Programming Software**: PNOZ Configurator 2.0.0
- **Device Runtime Software**: None

For a list of the devices used for conformance testing, see “Conformance Testing.”

Link Characteristics

To establish communication, you must use links with the following specifications:

- **Device Communication Port**: PNOZ m1p serial port (X4)
- **Physical Protocol**: RS232
- **Logic Protocol**: Proprietary
- **Specific PC Board**: None

Driver Characteristics

The PNOZdriver is composed of the following files:

- **PNOZ.INI**: Internal driver file. *You must not modify this file*.
- **PNOZ.MSG**: Internal driver file containing error messages for each error code. *You must not modify this file*.
- **PNOZ.PDF**: Document providing detailed information about the PNOZ driver.
- **PNOZ.DLL**: Compiled driver.

> **Notes:**
> - All of the preceding files are installed in the **`/DRV`** subdirectory of the Studio installation directory.
> - You must use Adobe Acrobat® Reader™ (provided on the Studio installation CD-ROM) to view the *PNOZ.PDF* document.

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

- Windows NT/2K/XP
- Windows CE

For a list of the operating systems used for conformance testing, see “Conformance Testing” on page 4.

The PNOZ driver supports the following commands:

Command Type	Write	Read	Bit	Integer	DWord	String	Present Version PLC	Old Version PLC
40h – read_version (PNOZ Version)	–	•	–	–	•	–	•	•
41h – read_io (Input / Output)	–	•	•	–	–	–	•	•
43h – read_led (LED data)	–	•	–	•	–	–	•	•
44h - read_status (group message)	–	•	–	•	–	–	•	–
50h – read_table	–	•	–	•	–	–	•	–
2Ch – read_status_virtual_io	–	•	–		•	–	•	–
2Dh – read_diagnostic	–	•	–	•	–	–	•	–
22h – read_stack (Error Stack)	–	•	–	–	–	–	–	•
driver_settings (Set path string)	•	•	–	–	–	•	–	•
14h – write_virtual_imputs	•	–	–	–	•	•	•	–

Conformance Testing

The following hardware/software was used for conformance testing:

- **Driver Configuration**:
  - **PLC Program**: None
  - **Baud Rate**: 19200
  - **Data Bits**: 8
  - **Stop Bits**: 2
  - **Parity**: Even
  - **COM Port**: COM1
- **Cable**: Serial cable

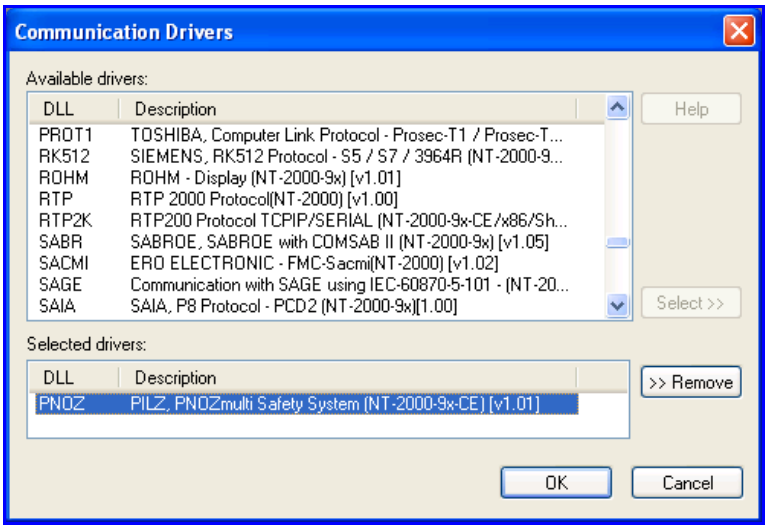
Driver Version	Studio Version	Operating System	Equipment
1.02	6.1+SP1	WinXP+SP1 WinCEv3.00	PNOZ m1p

Installing the Driver

When you install Studio version 5.1 or higher, all of the communication drivers are installed automatically. You must select the driver that is appropriate for the application you are using.

Perform the following steps to select the driver from within the application:

1. Open Studio from the **Start** menu.
2. From the Studio main menu bar, select **File** → **Open Project** to open your application.
3. Select **Insert** → **Driver** from the main menu bar to open the *Communication Drivers* dialog.
4. Select the **PNOZ** driver from the *Available Drivers* list, and then click the **Select** button:



Communication Drivers Dialog

5. When the **PNOZ** driver displays in the **Selected Drivers** list, click the **OK** button to close the dialog.

> **Note:**
> It is not necessary to install any other software on your computer to enable communication between the host and the device. However, to download the custom program to your device, you must install the PNOZmulti Configurator software. Consult your PNOZ programmer software documentation for installation instructions.

> **Attention:**
> For safety reasons, you must use special precautions when installing the physical hardware. Consult the hardware manufacturer's documentation for specific instructions in this area.

Configuring the Driver

After opening Studio and selecting the PNOZ driver, you must configure the driver. Configuring the PNOZ driver is done in two parts:

- Specifying communication parameters
- Defining tags and controls in the *STANDARD DRIVER SHEET*s (or Communication tables)

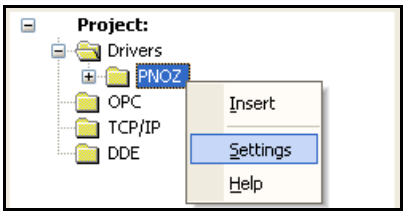
  Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header** and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header** and **Address** fields only.

> **Note:**
> For a detailed description of the Studio *STANDARD DRIVER SHEET*s, and information about configuring the standard fields, review the product's *Technical Reference Manual*.

Setting the Communication Parameters

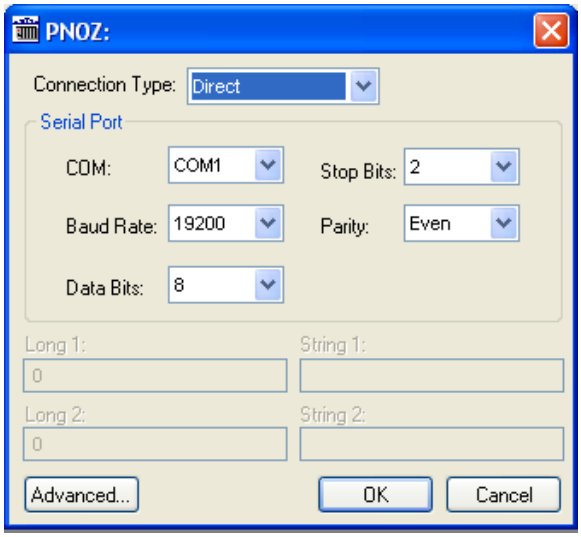
Use the following steps to configure the communication parameters, which are valid for all *Driver* worksheets configured in the system:

1. From the Studio development environment, select the **Comm** tab located below the *Workspace* pane.
2. Click on the *Drivers* folder in the *Workspace* pane to expand the folder.
3. Right-click on the *PNOZ* subfolder and when the pop-up menu displays, select the **Settings** option:



Select Settings from the Pop-Up Menu

The *PNOZ: Communication Parameters* dialog displays:



Communication Parameters Dialog

4. Click the **Advanced** button on the *Communication Parameters* dialog to open the *Advanced Settings* dialog and configure the settings that are necessary.

> **Notes:**
> - Do not change any of the other *Advanced* parameters at this time. You can consult the Studio *Technical Reference Manual* for information about configuring these parameters for future reference.
> - Generally, you must change the *Advanced* parameter settings if you are using a DCE (Data Communication Equipment) converter (232/485 for example), modem and so forth between the PC, driver and the host. You must be familiar with the DCE specifications before adjusting these configuration parameters.

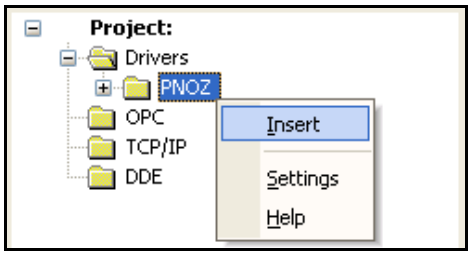
Configuring the Driver Worksheets

This section explains how to configure the *STANDARD DRIVER SHEETs* (or Communication tables) to associate application tags with the device addresses. You can configure multiple *Driver* worksheets — each of which is divided into a *Header* section and *Body* section.

Configuring the *STANDARD DRIVER SHEET*

Use the following steps to create a new *STANDARD DRIVER SHEET*:

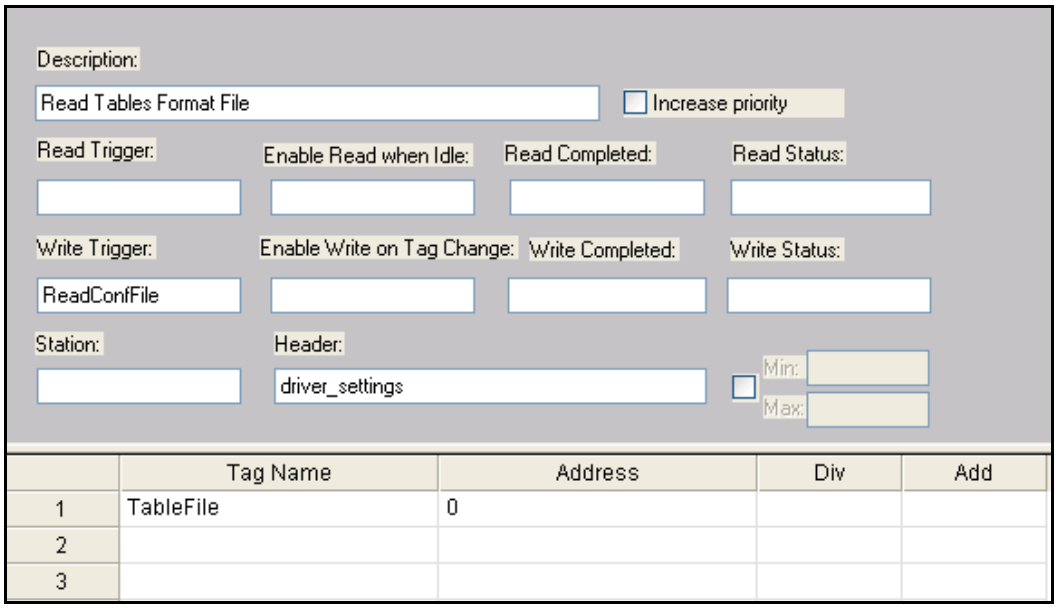
1. From the Studio development environment, select the **Comm** tab, located below the *Workspace* pane.
2. In the *Workspace* pane, expand the *Drivers* folder and right-click the *PNOZ* subfolder.
3. When the pop-up menu displays, select the **Insert** option:



Inserting a New Worksheet

> **Note:**
> To optimize communication and ensure better system performance, you must tie the tags in different driver worksheets to the events that trigger communication between each tag group and the period in which each tag group must be read or written. Also, we recommend configuring the communication addresses in sequential blocks to improve performance.

The *STANDARD DRIVER SHEET* displays (similar to the following figure):



STANDARD DRIVER SHEET

In general, all parameters on the *Driver* worksheet (except the **Station**, **Header** and **Address** fields) are standard for all communication drivers, but they will not be discussed in this document. For detailed information about configuring the standard parameters, consult the *Studio Technical Reference Manual*.

4. Use the following information to complete the **Station**, **Header** and **Address** fields on this worksheet:
   – **Station** field: Not used.
   – **Header** field: Use the information in the following table to define the type of variables that will be read from or written to the device (default value is **`read_version`**).

     These variables must comply with the following syntax:

     <*Command*> (for example: **`read_version`**)

     Where:

     Command is the data type (**`40`** or **`read_version`**; **`41`** or **`read_io`**; **`43`** or **`read_led`**; **`44 or read_status; 50`** or **`read_table; 2C`** or **`read_status_virtual_io; 22`** or **`read_stack; 14 or write_virtual_inputs`** and **`driver_settings).`**

     `<Command>:<ElementID>`

     `Where:`

     `Command` `is the` **`2D`** `or` **`read_diagnostic.`**

     `ElementID` `is the requested` **`Element ID (1..100).`**

     `<Command>:<Table>.<Segment>`

     `Where:`

     `Command` `is the` **`50`** `or` **`read_table.`**

     `Table` `is the table number.`

     `Segment` `is the segment number.`

After you edit the **Header** field, Studio checks the syntax to determine if it is valid. If the syntax is incorrect, Studio automatically inserts the default value in the **Header** field.

Also, you can type a tag string in brackets **`{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.

The following table lists all of the data types and address ranges that are valid for the PNOZ driver.

Header Field Information		
Data Types	**Sample Syntax**	**Comments**
40 or "read_version"	read_version	**Version**: Read the information versions about the PNOZmulti device.
41 or "read_io"	read_io	**Input and Output**: Read inputs and outputs data from the PNOZmulti device.
43 or "read_led"	read_led	**LED**: Read LED status.
44 or "read_status"	read_status	**Status Scan**: Read simplified status scan from the PNOZmulti device.
50 or "read_table"	read_table	**Data in table form**: Read data, in table form, from the PNOZmulti device.
2C or "read_status_virtual_io"	read_status_virtual_io	**Status Virtual IO**: Read status of virtual inputs and outputs, from the PNOZmulti device.
2D or "read_diagnostic"	read_diagnostic	**Diagnostic Word**: Read diagnostic word, from the PNOZmulti device.
22 or "read_stack"	read_stack	**Stack Error Messages**: Read the Stack Error Messages and save in the CSV file. **Obs.: Execute first the "driver_settings" command to configuration of parameters.**
"driver_settings"	driver_settings	**Driver Settings**: Read and write the new value in the internal variables to driver settings (PNOZ File , PNOZ File Version, StackErrorMessages File, Remedy File, CVS File, include Full Text and include Remedy). **Obs.: Execute this command before of use the "read_stack" (22) command.**
14 or write_virtual_inputs	write_virtual_inputs	**Virtual Inputs**: Write 24 virtual inputs, to the PNOZmulti device.

– **Address** field: Use this field to associate each tag to its respective device address.

Type the tag from your application database into the **Tag Name** column. This tag will receive values from or send values to an address on the device. The address must comply with the following syntax:

`<Address>.[Bit]` (for example: **`10`**, **`0`**, **`10.5`**).

Where:

* **Address** is the parameter number to read from or write to the device, and it is different for any command.
* **Bit** (*optional parameter used for 41 types only*) is the bit number to be read from the device.

Address Field Information		
Header	**Valid Address**	**Comments**
40 or "read_version"	0.<bit> to 16.<bit> (cont.)	0 – Product Number 1 – Device Number 2 - Serial Number 3 – Check sum user program 4 – Check sum user data

Address Field Information		
Header	**Valid Address**	**Comments**
		5 – Creation date user program (day) 6 – Creation date user program (month) 7 – Creation date user program (year) 8 – Hardware registry module left No expansion module: 00 Virtual inputs and outputs: 40 9 .. 16 – Hardware registry module right No expansion module: 00 Expansion modules: PNOZ mi1p: 08 PNOZ mo1p: 18 PNOZ mo2p: 10 PNOZ mo4p: 28 PNOZ mc1p: 20 PNOZ ms1p: 88
41 or "read_io"	0.<bit:> to 9.<bit>	**0 – Base Unit (Input and Output)** <bit> - 0 to 39 0 to 19 - **Inputs 0 to 19** 20 to 23 – Reserved 24 to 27 – **Outputs 0 to 3** 28 to 31 – Reserved 32 and 33 – **Output 4 and 5** 34 to 39 – Reserved **1 to 9 – expansion modules** <bit> - 0 to 15
43 or "read_led"	0.<bit> to 31.<bit>	**0 – Shows the system's operating status (Start, Run, STOP)** Value: 18 - START 83 - RUN 162 – STOP **1 to 13 – Shows the LED status (Run, DIAG, FAULT)** Value: 0 - Diag OFF 255 - Diag ON 48 - Diag Flashing **14 to 26 – Show the input LEDs** <bit> - 0 to 7 Value: 0 or 1 – Flashing / not Flashing **27 to 29 – Show the status of CI, CO and OA0 LEDs** Value: 00 or 255 – Off / ON **30 to 31 – Reserved**
44 or read_status	0.<bit> to 6.<bit>	Value: 0 or 1 0 – Error at an output 1 – Error at an input 2 – LED FAULT is lit/flashes 3 – LED DIAG is lit/flashes 4 – LED RUN is lit 5 – At least one input signal has changed since the last 0x44 request 6 – At least one output signal has changed since the last 0x44 request
50 or read_table	0 .. 12	0..12 - Byte n, contains the data from Segment x of Table y (specified in the header field).

Address Field Information		
Header	**Valid Address**	**Comments**
2C or read_status_virtual	0..1	**0 – status of 24 virtual inputs.** Each bit (0 to 23) is a status of virtual inputs **1 – status of 24 virtual outputs.** Each bit (0 to 23) is a status of virtual outputs
2D or read_diagnostic	0	**0 -** Diagnostic word for a specific Element ID (specified in the header field).
22 or "read_stack"		**Not used address field information**
"driver_settings"	0 to 6	**0 - PNOZ File.** String Tag that contains the PNOZ file (*.mpnoz) name and full path. **1 - PNOZ File Version (Read only)** Version of PNOZ file format. **2 - Error Stack Message path and file name (default: ErrorStackMessagesFile.txt)** String Tag that contains the Error Stack Message file (*.txt) name and full path – this file can be found in the Pilz Programming software folder. **3 - Remedy path and file name (default: RemedyFile.txt)** String Tag that contains the Remedy file (*.txt) name and full path – this file can be found in the Pilz Programming software folder. **4 - CSV path and file name (default: InduErrorStackMessagesFile.csv)** String Tag that contains the CVS file (*.txt) name and full path – this file will be created and modified by the driver when the command **Read Trigger** of a worksheet with the Header "read_stack" is executed. **5 – Include Full Text** A column with the Full Text is included in the CSV file when it is enabled. **6 – Include Remedy** A column with the Remedy text is included in the CSV file when it is enabled. The driver will get the Remedy text from this Remedy file.
14 or "write_virtual_inputs"	0	**0 – Set 24 virtual_inputs** **Each bit (0 ..23) is a virtual input.**

Address Configuration Sample		
Device Address	**Header Field**	**Address Field**
Product Number	40	0
Serial Number	read_version	2
Input 0 from Base Unit	41	0.0
Input 5 from Base Unit	read_io	0.5
Input 19 from Base Unit	41	0.19
Output 0 from Base Unit	read_io	0.24
Output 3 from Base Unit	41	0.27
Output 4 from Base Unit	read_io	0.32
Output 5 from Base Unit	41	0.33
System´s operating status	read_led	0
LED 1 Status	43	1
LED 5 Status	read_led	5
Input 0 LED Status	43	14.0
Error Stack Messages	Read_stack	0
Set PNOZ file	driver_settings	0

Set Remedy file name	SETSTR	2

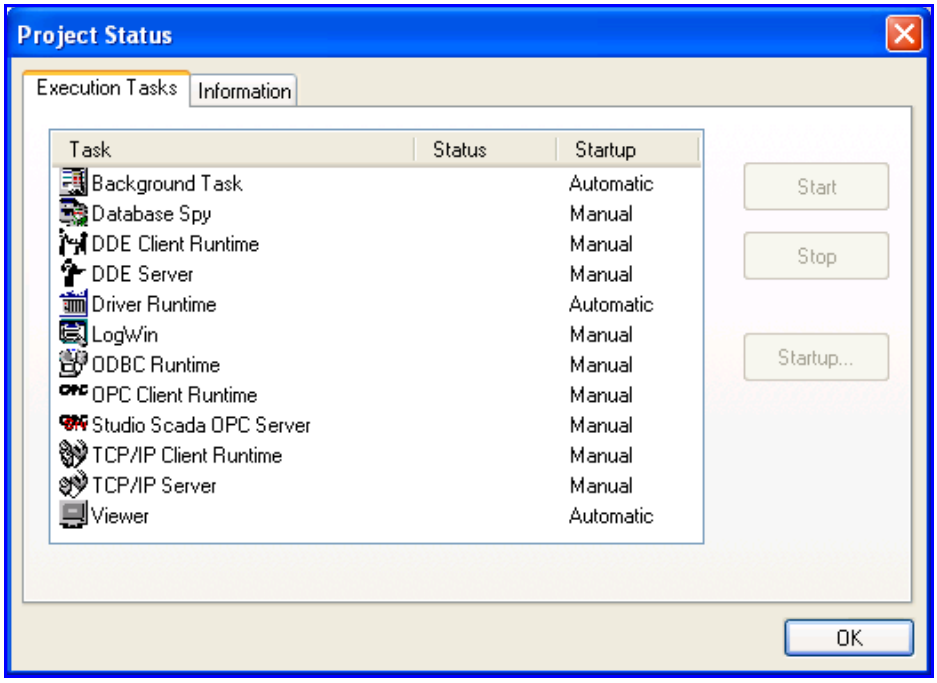
Example of a script to capture the File Names to String tags and load in the "driver_settings" Worksheet **	
Tag	**Expression**
SysAppPath	If (GetOS() = 3, InfoAppDir(), InfoAppDir() + "\")
PNOZ_File	SysAppPath + "test_indu.mpnoz"
ErrorStackMessagesFile	SysAppPath + "ErrorStackMessagesFile_en.txt"
PNOZ_File_Version	""
RemedyFile	SysAppPath + "RemedyFile_en.txt"
InduErrorStackMessagesFile	SysAppPath + "InduErrorStackMessagesFile.csv"
Include_Full_Text	1
Include_Remedy	1
Obs. Use this command only if the file .mpnoz is in the same directory of your application.	

Executing the Driver

After adding the PNOZ driver to a project, Studio sets the project to execute the driver automatically when you start the run-time environment.

To verify that the driver run-time task is enabled and will start correctly, perform the following steps:

1. Select **Project** → **Status** from the main menu bar.
   The *Project Status* dialog displays:



Project Status Dialog

2. Verify that the *Driver Runtime* task is set to **Automatic**.
   – If the setting is correct, click **OK** to close the dialog.
   – If the **Driver Runtime** task is set to **Manual**, select the **Driver Runtime** line. When the **Startup** button becomes active, click the button to toggle the *Startup* mode to **Automatic**.
3. Click **OK** to close the *Project Status* dialog.
4. Start the application to run the driver.

Troubleshooting

If the PNOZ driver fails to communicate with the device, the tag you configured for the **Read Status** or **Write Status** fields 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	Communication without problems.	None required
10	Invalid Command	Write trigger was tried in the read-only commands.	Change the configured command, or try only the Read trigger.
20	Invalid Command Number	The command configured the Header is not valid to this driver.	Check the possible commands in this document, and modify the configuration in the driver worksheet.
30	No Error Stack Message received from device	The invalid message was received from the device.	Check the communication parameters for the device and the Studio software.
40	Negative Ack	Error in the communication Ack action. Specialized use in conjunction with programming commands.	Check the communication parameters for the device and the Studio software.
50	Invalid driver settings	Error invalid driver settings	Check the specifications of commands
60	Error read table of file PNOZ	Error in the read of information in the file specified in the "driver settings" command.	Check the file name specified in the "driver settings" command.
-15	Timeout Start Message	▪ Disconnected cables ▪ PLC is turned off, in stop mode, or in error mode ▪ Wrong station number ▪ Wrong RTS/CTS control settings	▪ Check cable wiring. ▪ Check the PLC state – it must be RUN. ▪ Check the station number. ▪ Check the configuration. See *Studio Technical Reference Manual* for information about valid RTS/CTS configurations.
-17	Timeout between rx char	▪ PLC in stop mode or in error mode ▪ Wrong station number ▪ Wrong parity ▪ Wrong RTS/CTS configuration settings	▪ Check cable wiring. ▪ Check the PLC state – it must be RUN. ▪ Check the station number. ▪ Check the configuration. See *Studio Technical Reference Manual* for information about valid RTS/CTS configurations.

⇨ **Tip:**

You can verify communication status using the Studio development environment *Output* window (*LogWin* module). To establish an event log for **Field Read Commands**, **Field Write Commands**, and **Serial Communication,** right-click in the *Output* window. When the pop-up menu displays, select the option to set the log events. If you are testing a Windows CE target, you can use the *Remote LogWin* of Studio (**Tools** → **Remote LogWin**) to get the log events from the target unit remotely.

If you are unable to establish communication with the PLC, try to establish communication between the PLC Programming Tool and the PLC. Quite frequently, communication is not possible because you have a hardware or cable problem, or a PLC configuration error. After successfully establishing communication between the device's Programming Tool and the PLC, you can retest the supervisory driver.

To test communication with Studio, we recommend using the sample application provided rather than your new application.

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**.
- **Studio Version**: To find this information, select **Help** → **About.**
- **Driver Version**: To find this information, read the full description of the driver on the *Communication Drivers* dialog.
- **Communication Log**: Displays in the Studio *Output* window (or *LogWin* window) when the driver is running. Be sure to enable the **Field Read Commands**, **Field Write Commands** and **Serial Communication** for the *LogWin* window.
- **Device Model** and **Boards**: Consult the hardware manufacturer's documentation for this information.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of changes
A	1.00	Roberto Vigiani Jr	Nov/19/2003	Initial revision.
B	1.01	Fábio H.Y. Komura	Jun/28/2004	▪ Included PNOZ table version 1.0.9 ▪ Fixed problem when opening file with Ctrl-Z character (EOF).
C	1.02	Jorge Luiz	Jul/27/2006	▪ Include commands (0x14,0x2c,0x2d,0x44 and 0x50). ▪ Fixed problem with command 0x14.
C	1.03	Lourenço Teodoro	Dec/16/2008	▪ Updated driver version, no changes in the contents.