Communication Driver XRC

Slave Driver for serial communication with Motoman XRC devices using RS-232

Index

1 Introduction

The XRC driver enables communication between Studio system and some of the Motoman devices using their BSC compliant protocol by RS 232, in accordance with the characteristics covered in this document.

This document contains 8 parts, as follow:

- Introduction: Provides an overview of the driver documentation.
- General characteristics: Provides information necessary to identify all the required components (hardware and software) necessary to implement the communication and global characteristics about the communication.
- Installation: Explains the procedures that must be followed to install the software and hardware required for the communication.
- Driver configuration: Provides the required information to configure the communication driver such as the different permutations for configuration and its default values.
- Execution: Explain the steps to test whether the driver was correctly installed and configured.
- Troubleshooting: Supplies a list of the most common error codes for this protocol and the procedures to fix them.
- Application Sample: Provides a sample application for testing the configuration the driver.
- History of versions: Provides a log of all the modifications done in driver.

☞ Note: This document presumes that the user has read the chapter *Driver Configuration* of the Studio's Technical reference manual.

2 General Characteristics

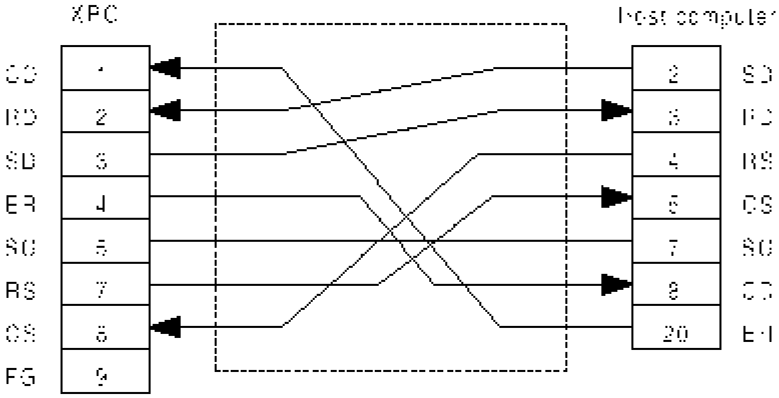
2.1 Device Characteristics

- Manufacturer: Motoman
- Compatible Equipment
  - XRC

☞ Note: Please refers to section 2.4 to see the Equipment used in the standard conformance tests for this driver.

2.2 Link Characteristics

- Device communication port: RS232 port
- Physical protocol: RS232/RS485
- Logic protocol: BSC compliant Slave Protocol
- Device Runtime software: None
- Specific PC Board: None
- Adapters / Converters: None.
- Cable Wiring: We can see it on the following picture bellow



2.3 Driver Characteristics

- Operating System:
  - Windows 9x
  - Windows 2000
  - Windows NT
  - Windows CE

☞ Note: Please refer to section 2.4 to see the Operating System used in the conformance tests for this driver.

The driver is composed of the following files:

- XRC.INI: Internal file of the driver, it should not be modified by the user.
- XRC.MSG: This file contains the error messages for each error code. It is an internal file of the driver, the user should not modify it.
- XRC.PDF: This document provides detailed documentation about the driver.
- XRC.DLL: This is the compiled library for the driver.

☞ Note: All the files above must to be in the subdirectory /DRV of the Studio's installation directory.

2.4 Information about conformance testing

- Equipment: XRC

Configuration:

Baud Rate: 19200
Protocol: BSC compliant Slave Protocol
Data Bits: 8
Stop Bits: 1
Parity: Even
COM port: COM1

- Cable: According link specification, section 2.2.
- Operating System (development): Windows NT 4.0 + Service pack 5
- Operating System (target): Windows NT 4.0 + Service Pack 5; Windows CE v2.11
- Studio Version: 3.0
- Driver version: 1.00

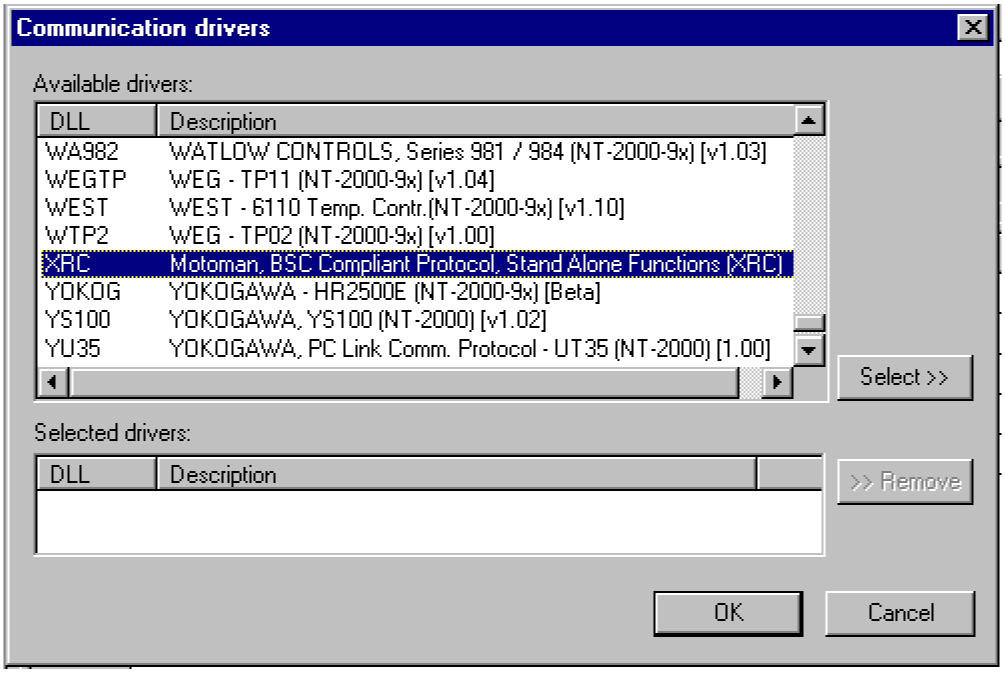
3 Installation

3.1 *Installing the Driver*

When you install the Studio v3.0 or higher, the communication drivers are already installed. You need now to select the driver at the applications where it will be used.

The steps to select the driver inside an application are:

1. Execute the Studio and select the proper application.
2. Select the menu *Insert + Driver...*
3. In the column **Available Drivers**, select the **XRC Driver** and push the button **ADD>>>** (the driver XRC must appear in the column **Selected Drivers**).
4. Press **OK**.



Available drivers:

DLL	Description
WA982	WATLOW CONTROLS, Series 981 / 984 (NT-2000-9x) [v1.03]
WEGTP	WEG - TP11 (NT-2000-9x) [v1.04]
WEST	WEST - 6110 Temp. Contr.(NT-2000-9x) [v1.10]
WTP2	WEG - TP02 (NT-2000-9x) [v1.00]
XRC	Motoman, BSC Compliant Protocol, Stand Alone Functions (XRC)
YOKOG	YOKOGAWA - HR2500E (NT-2000-9x) [Beta]
YS100	YOKOGAWA, YS100 (NT-2000) [v1.02]
YU35	YOKOGAWA, PC Link Comm. Protocol - UT35 (NT-2000) [1.00]

Selected drivers:

DLL	Description

3.2 *Other software requirements*

It is not necessary to install any other software in the PC to enable the communication between the host and the Device.

☞ Note: Special precautions must be taken when installing the physical hardware. Refer to the hardware manufacturer documentation for specific instructions in this area.

4 Driver Configuration

After the driver is installed and selected in the Studio (see section 3.1), you should proceed to the driver configuration.

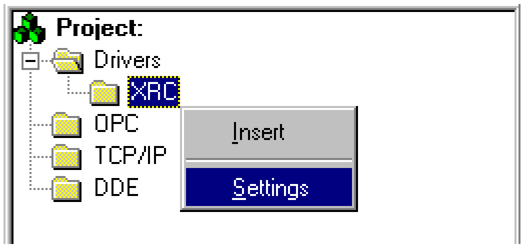
The driver configuration is two parts:

The Settings or Communication parameters, it is only one configuration to the whole driver, them you have the communication tables or Driver Worksheets, where the communication tags are defined.

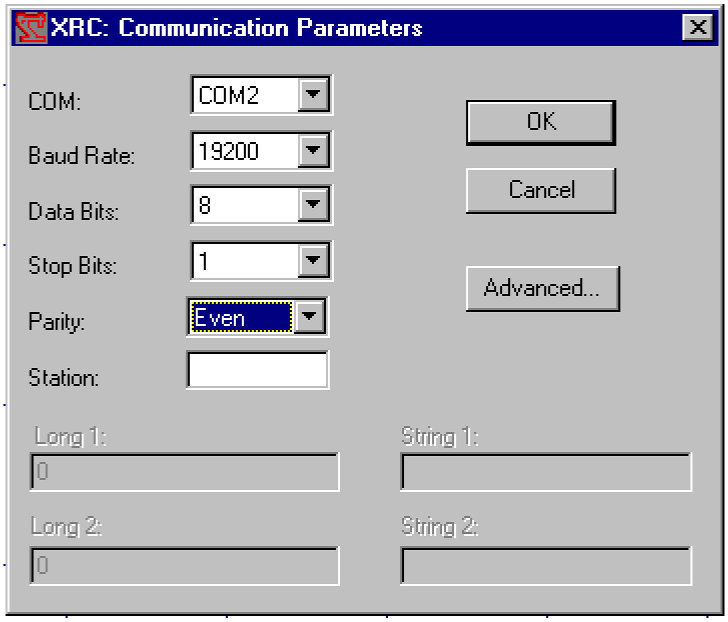
4.1 Settings - Communication Parameters

These parameters are valid for all driver worksheets configured in the system. To open the window for configuring the **Communication parameters**, follow these steps:

1. In the **Workspace** of the Studio environment, select the **Comm** table.
2. Expand the folder **Drivers** and select the subfolder **XRC**.
3. Right click on the **XRC** subfolder and select the option **Settings**.



When selecting the Settings, there is the following dialog to configure:



COM: COM2

Baud Rate: 19200

Data Bits: 8

Stop Bits: 1

Parity: Even

Station:

Long 1: 0

String 1:

Long 2: 0

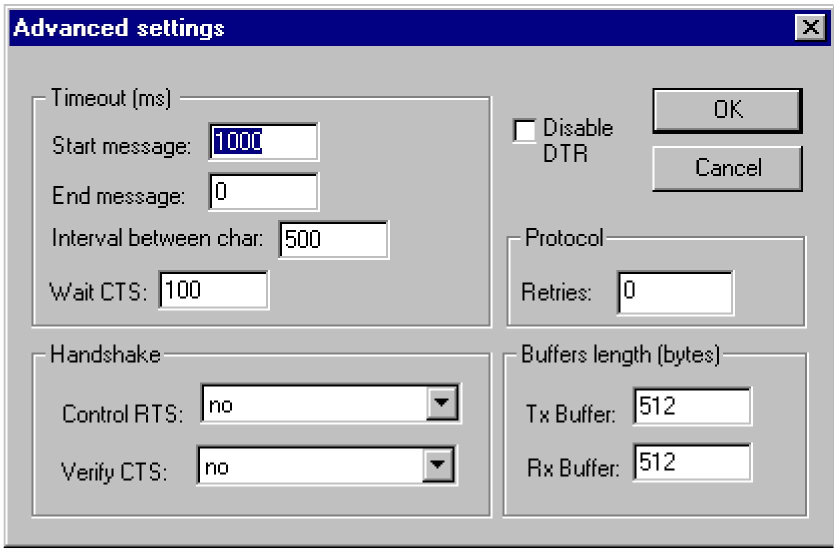
String 2:

Parameter	Default Value	Valid values	Description
COM	COM2	COM1 to COM8	Serial port of the PC used to communication with the device
Baud Rate	19200	110 to 57600bps	Communication rate of data

Data Bits	8	5 to 8	Number of data bits used in the protocol
Stop Bits	1	1or 2	Number of stop bits used in the protocol
Parity	Even	even, odd, none, space or mark	Parity of the protocol
Station	-	-	This field must to be filled with the value 0 (zero).

☞ Note: These Parameters must be just the same as the configured on the XRC device.

By clicking on the button **Advanced…** in the window **Communication Parameters,** you open additional communication parameters.



Advanced settings

Timeout (ms)

Start message: 1000

End message: 0

Interval between char: 500

Wait CTS: 100

□ Disable DTR

Protocol

Retries: 0

Handshake

Control RTS: no

Verify CTS: no

Buffers length (bytes)

Tx Buffer: 512

Rx Buffer: 512

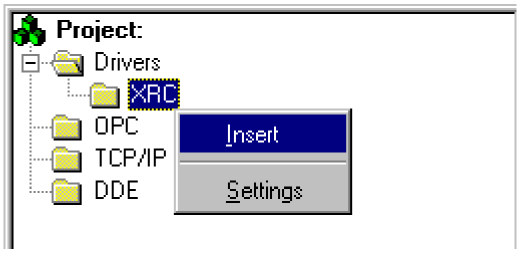
The Advanced setting parameters are explained at the Studio Technical Reference Manual, and you should keep the default values to all fields. Only the field described at the next table should be configured:

Parameter	Default Value	Valid values	Description
Control RTS	No	no, yes or yes + echo	Define if the handshake signal of RTS (Request to Send) is set before communication and if there is an echo in the communication. If we are using Windows 95 or CE with the right RS 232 – RS 485 Converter (without RTS Control), choose the option "**no**". **Important**: Wrong settings on this field will not let the driver work, having the Timeout waiting start a message error.

4.2 Driver Worksheet

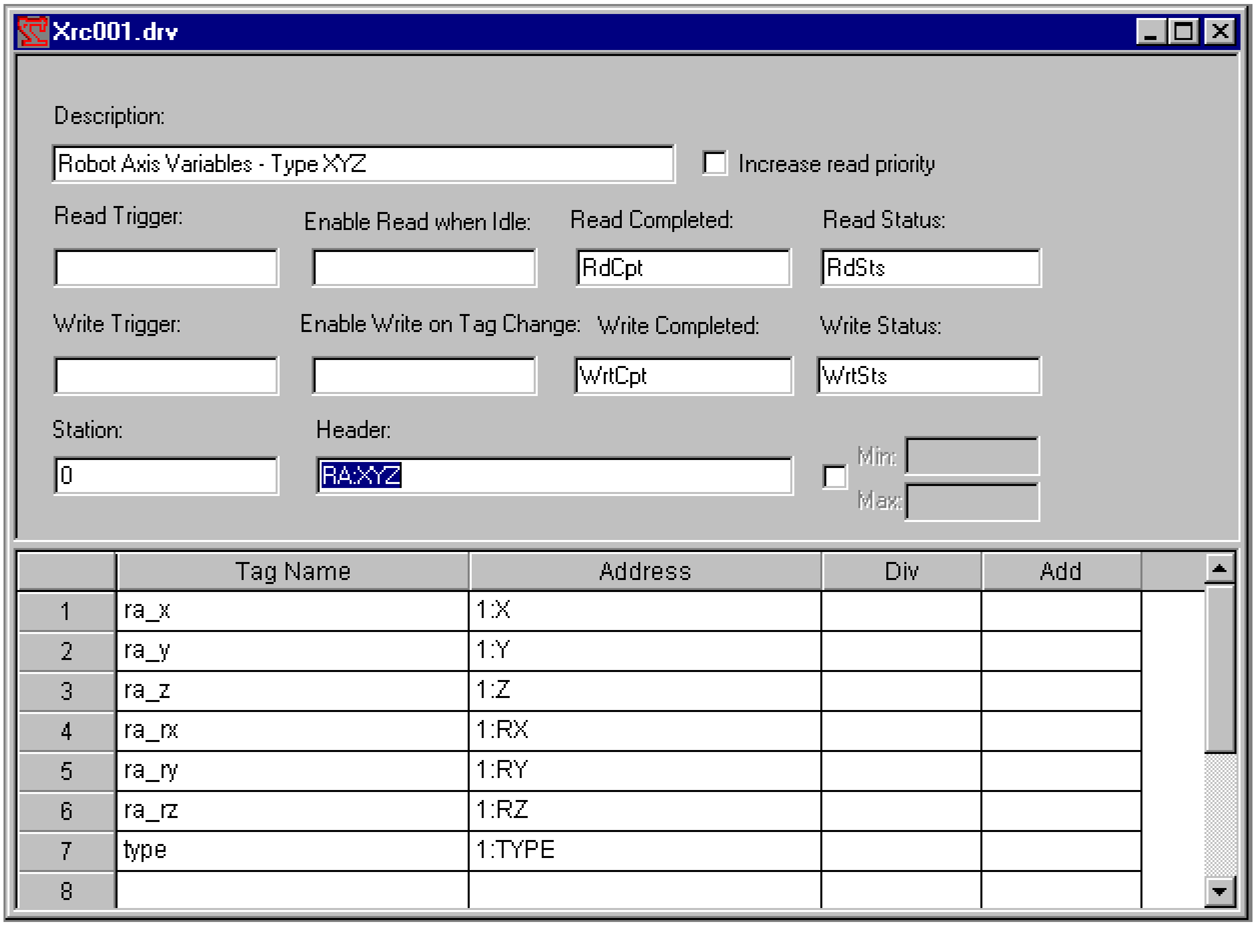
It is possible to configure many driver worksheets, each one will be composed of a Header and Body. To create a new driver worksheet, follow these steps:

1. In the **Workspace** of the Studio environment, select the table **Comm**.
2. Expand the folder **Drivers** and select the subfolder **XRC**.
3. Right click on the **XRC** subfolder and select the option **Insert**.



☞ Note: To optimize communication and ensure better performance for the system, it is important to tie the tags in different driver sheets according to the events that must trigger the communication of each group of tags and the periodicity for which each group of tags must be written or read. In addition, it is recommended to configure the addresses of communication in sequential blocks.

When creating a communication table, you have the following window:


Xrc001.drv

Description: Robot Axis Variables - Type XYZ

□ Increase read priority

Read Trigger:	Enable Read when Idle:	Read Completed:	Read Status:
		RdCpt	RdSts
Write Trigger:	Enable Write on Tag Change:	Write Completed:	Write Status:
		WrtCpt	WrtSts

Station: 0

Header: RA:XYZ

□ Min: Max:

	Tag Name	Address	Div	Add
1	ra_x	1:X		
2	ra_y	1:Y		
3	ra_z	1:Z		
4	ra_rx	1:RX		
5	ra_ry	1:RY		
6	ra_rz	1:RZ		
7	type	1:TYPE		
8				

All entries at the Driver Worksheet, exception by the **Station**, **Header** and **Address** are standard to all communication drivers. You should refer to Studio Communication Driver documentation about the configuration of the standard fields. This document describes the Station, Header and Address fields, which are specific to each communication driver.

☞ Note: In this protocol, the driver is a slave device, so the fields **Read Trigger**, **Enable Read when Idle**, **Write Trigger** and **Enable Write on Tag Change** are not used in this particular case.

4.3 Station and Header configuration

Parameter	Default Value	Valid values	Description
Station	-	None	
Header	I	Vide next table	Defines the type of variable to be read or written from or to the device.

The **Header** field defines the type of variables that will be read or written from or to the device. After editing the field **Header**, the system will check if it is valid or not. If the syntax were incorrect, the default value (W:0) will be automatically placed in this field.

You can type Tag between curly brackets into this field, but be sure that the Tag's value is correct, with the right syntax, or you will get the Invalid Header error. The right syntax, both for the field typing and Tag value is described bellow:

Information regarding the parameter "Header"			
Type	**Sample of syntax**	**Valid range of initial Address**	**Comment**
Byte	B	-	Save/Load variables type byte from/to device
Integer	I	-	Save/Load variables type integer from/to device
Double	D	-	Save/Load variables type double precision from/to device
Real	R	-	Save/Load variables type real from/to device
Robot Axis Position XYZ type	RA:XYZ	-	Save/Load robot axis position type xyz from/to device
Robot Axis Position Pulse type	RA:PULSE	-	Save/Load robot axis position type pulse from/to device
External Axis Position XYZ type	EA:XYZ	-	Save/Load external axis position type xyz from/to device
External Axis Position XYZ type	EA:PULSE	-	Save/Load external axis position type pulse from/to device
JOB Files	JOB	-	Save/Load JOB files from/to device

4.4 Address Configuration

The body of the driver worksheet allows you to associate each tag to its respective address in the device. In the column **Tag Name,** you must type the tag from your application database. This tag will receive or send values from or to an address on the device.

The address cells complies to the following two possible syntaxes:

<INDEX>:<PARAMETER>
<INDEX>
<DIRECTORY>

INDEX: the **index** is a number that the device send first to the driver configuring the position of the variable into the internal table of the device. This number must be always configured for variables operation.

PARAMETER: The **parameter** is an identifier used only on position variables to distinguish the various different type of position variables.

DIRECTORY: The **directory** is the path name of the job file used to save/load the job files from/to device.

Sample of Addressing Configuration		
Header Field	**Address Field**	**Comment**
B	13	Save/Load byte variable on index 13
I	1	Save/Load integer variable on index 1
D	32	Save/Load double variable on index 32
R	15	Save/Load real variable on index 15
RA:XYZ	2:X	Save/Load ‘X’ parameter of the robot axis on index 2
RA:PULSE	3:U	Save/Load ‘U’ parameter of the robot axis on index 3
EA:XYZ	5:Z	Save/Load ‘Z’ parameter of the external axis on index 5
EA:PULSE	4:1	Save/Load ‘1’ parameter of the external axis on index 4

☞ Note: To load position variables to the device, the driver worksheet must have configured all parameters of the position variable. For example, to robot axis, the driver worksheet must have the parameters X, Y, Z, RX, RY, RZ and TYPE configured, otherwise, the driver will generate an error.

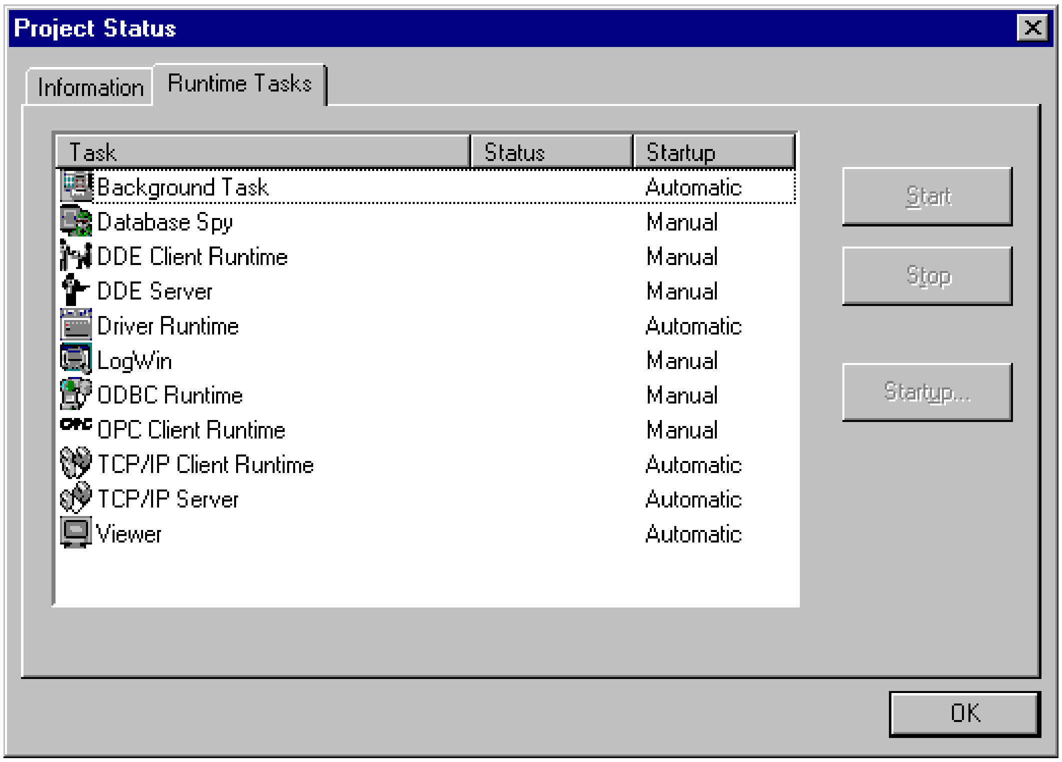
4.5 Device Configuration

The XRC device was configured with:

Baud Rate: 19200
Data bits: 8
Stop bits: 1
Parity: Even

5 Execution

- When installing the driver, it is automatically selected to execute when you start-up the Runtime Environment. To verify the if the driver is correctly enabled to start, use the menu option **Project** + **Status…,** and verify the task Driver Runtime



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
TCP/IP Client Runtime		Automatic
TCP/IP Server		Automatic
Viewer		Automatic

6 Troubleshooting

After each attempt to communicate using this driver, the tag configured in the field **Read Status** or **Write Status** will receive the error code regarding the kind of failure that occurred. The error messages are:

Error Code	Description (*)	Possible causes	Procedure to solve
0	OK	Communication without problems	-
3	Invalid Header Field	Incompatible Header Entry	Check on the driver configuration's worksheet that the field header is properly configured as described in section 4.3.
4	Invalid Address Field	Incompatible Address Entry	Check on the driver configuration's worksheet that the field Address is properly configured as described in section 4.3 and 4.4.
8	Invalid Command	The fields **Read Trigger**, **Enable Read when Idle**, **Write Trigger** and **Enable Write on Tag Change** are being used	These fields are not used. Delete tags configured in these fields. See note on section 4.2
9	Memory Allocation Error	The driver couldn't allocate memory to store the variables	Check the minimum system requirements to run the application and the integrity of the memory in the system
10	Cannot find the specific driver worksheet	The device required a specific variable but there isn't any driver worksheet with this information. For example, the device requires byte variable, but there isn't any driver worksheet with byte variables	Create a driver worksheet with the required information
11	Invalid File Name	The device required a job operation (save/load) but the driver received an incorrect communication frame (file name without CR)	Make sure that the connection cable is properly attached in both devices and check the protocol configured is a BSC compliant protocol
12	Cannot open the job file	The device required a job operation (save/load) but the driver couldn't open the required job file	Make sure that the file exist and is error free (load) or there is space enough to store the file in the system (save)
13	Cannot write data in the job file	The device required a save job operation but the driver couldn't write data in the file	Make sure that the file is error free and there is space enough to store the file in the system
15	Cannot find the index in the driver worksheet	The device required save/load variable operation but the driver couldn't find the required index	Configure the required index in the driver worksheet
16	Invalid required variable	The device required a save/load variable operation with an invalid variable type. For example, save float (F) variable	Make sure that the connection cable is properly attached in both devices and check the protocol configured is a BSC compliant protocol
17	Cannot fill all parameters required by load operation	The device required a load position variable operation, but neither all required parameters are configured. For example, load RA:XYZ, but only X and Y parameters are configured in the driver worksheet	Check if all required parameters are configured in the driver worksheet

18	Error on protocol procedure	The device required a load position variable operation, but neither all required parameters are configured. For example, load RA:XYZ, but only X and Y parameters are configured in the driver worksheet	Check if all required parameters are configured in the driver worksheet
-15	Timeout waiting start a message.	- Disconnected cables - PLC turned off, or in Stop or error mode - Wrong Station number - Wrong RTS/CTS control settings.	- Check the cable wiring - Check the PLC state. It must be RUN - Check the station number. - Check the right configuration. See on the section 2.2 the different RTS/CTS valid configurations.
-17	Timeout between rx char.	- PLC in stop or error mode - Wrong station number - Wrong parity - Wrong RTS/CTS configuration settings	- Check the cable wiring - Check the PLC state. It must be RUN - Check the station number. - Check the right configuration. See on the section 2.2 the different RTS/CTS valid configurations.

☞ Note: The results of the communication may be verified in the **output** Window of the Studio's environment. To set a log of events for **Field Read Commands**, **Field Write Commands** and **Serial Communication** click with the right button of the mouse on the output window and chose the option setting to select these log events. When testing under a Windows CE target, you can enable the log at the unit (Tools/Logwin) and verify the file celog.txt created at the target unit.

When you are not able to establish the communication with the PLC, you should first of all establish the communication between the PLC Programming Tool and the PLC. Very frequently the communication it is not possible due to a hardware or cable problem, or due an error or lack of configuration at the PLC. Only after the communication between the PLC Programming Software and the PLC is working fine, you can test again the supervisory.

When testing the communication with the Studio, you should first use the application sample described at item 7, instead of the new application that you are creating.

If is required to contact technical support, please have the following information available:

- Operating System (type and version): To find this information use the Tools/System Information option
- Project information: It is displayed using the option Project/Status from the Studio menu
- Driver version and communication log: Available from Studio Output when running the driver
- Device model and boards: please refer to hardware manufacture's documentation

7 Application Sample

The Studio contains a configured project to test the driver. It is strongly recommended to do some tests with this application before beginning the configuration of the customized project, for the follow reasons:

- To understand better the information covered in section 4 of this document.
- To verify that your configuration is working.
- To certify that the hardware used in the test (device + adapter + cable + PC) is in working conditions before beginning the configuration of the applications.

☞ Note: The Application Sample is not available for all drivers.

The Studio application is in the directory: /COMMUNICATION EXAMPLES/<Driver Name>

To perform the test, you need to follow these steps:

- Configure the device communication parameters using manufacturer programmer software..
- Open the application /COMMUNICATION EXAMPLES/<Driver Name>
- Execute the application
- To display the following screen with some information about the communication, please execute the Viewer module in the Studio.

☞ Note: The application for testing may be used like a maintenance screen for the custom application.

8 History of Versions

Version	By	Date	Description of changes
1.00	Sergio Andre Poon	10-Apr-2000	▪ First driver version ▪ Driver available for Windows CE
1.10	José L. Teodoro	25-Jul-2000	▪ Implemented Load Jobs