N2JC Communication Driver

Driver for Serial Communication
with N2 Devices

Contents

INTRODUCTION **2**

GENERAL INFORMATION **3**

DEVICE CHARACTERISTICS 3
LINK CHARACTERISTICS 3
DRIVER CHARACTERISTICS 3
CONFORMANCE TESTING 4

INSTALLING THE DRIVER **5**

CONFIGURING THE DRIVER **6**

SETTING THE COMMUNICATION PARAMETERS 6
CONFIGURING THE DRIVER WORKSHEETS 7

EXECUTING THE DRIVER **13**

TROUBLESHOOTING **14**

SAMPLE APPLICATION **16**

REVISION HISTORY **17**

Introduction

The N2JC driver enables communication between the Studio system and devices using the N2 protocol communicating over Serial, according to the specifications discussed in this document.

This document was designed to help you install, configure, and execute the N2JC driver to enable communication with these devices. The information in this document is organized as follows:

- **Introduction**: Provides an overview of the N2JC driver documentation.
- **General Information**: Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the N2JC driver.
- **Installing the Driver**: Explains how to install the N2JC driver.
- **Configuring the Driver**: Explains how to configure the N2JC 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 N2JC 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/CE environment. If you are unfamiliar with Windows NT/2000/XP/CE, 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 N2JC driver and the N2 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**: N2 devices (or any device using the N2 protocol communicating over Serial)
- **Compatible Equipment**:
  - Any device that is fully compatible with the N2 protocol

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**: RS-485
- **Physical Protocol**: RS-485
- **Logic Protocol**: N2
- **Specific PC Board**: None

Driver Characteristics

The N2JC driver is composed of the following files:

- **N2JC.INI**: Internal driver file. *You must not modify this file*.
- **N2JC.MSG**: Internal driver file containing error messages for each error code. *You must not modify this file*.
- **N2JC.PDF**: Document providing detailed information about the N2JC driver
- **N2JC.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 *N2JC.PDF* document.

You can use the N2JC 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 N2JC driver supports the following registers:

Register Type	Object	Attribute	Write	Read
AI (Analog Input)	0-255	1-14	–	•
BI (Binary Input)	0-255	1-4	–	•
AO (Analog Output)	0-255	1-5	•	•
BO (Binary Output)	0-255	1-7	•	•
BD (Binary Data)	0-255	1-2	•	•
ADI (Analog Data Integer)	0-255	1-2	•	•
ADF (Analog Data Float)	0-255	1-2	•	•

Conformance Testing

The following hardware/software was used for conformance testing:

- **Driver Configuration**:
  - **COM Port**: COM1
  - **Baud Rate**: 9600
  - **Data Bits**: 8
  - **Stop Bits**: 1
  - **Parity**: None

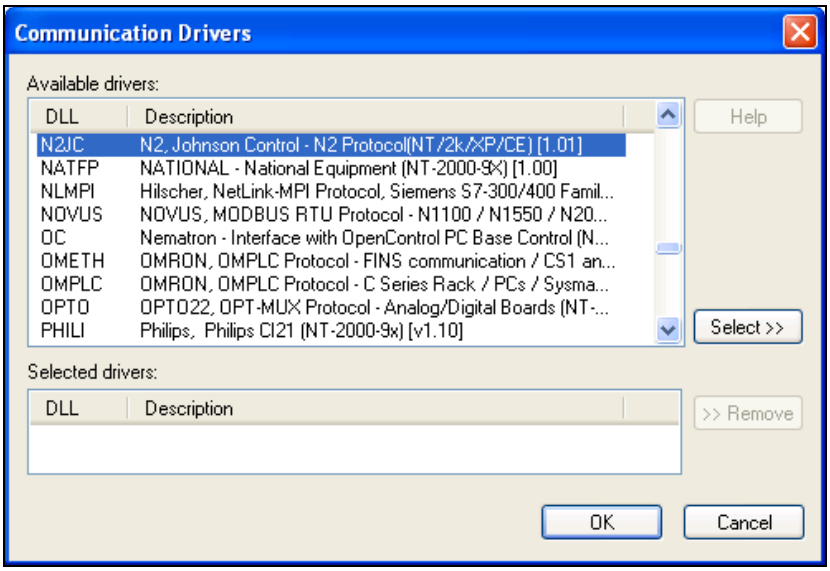
Driver Version	Studio Version	Operating System	Equipment
1.00	6.0	Windows XP	N2 device

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 **N2JC** driver from the *Available Drivers* list (as shown in the following figure), and then click the **Select** button.



Communication Drivers Dialog Box

5. When the **N2JC** driver displays in the **Selected Drivers** list, click the **OK** button to close the dialog.

> **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 N2JC driver, you must configure the driver. Configuring the N2JC driver is done in two parts:

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

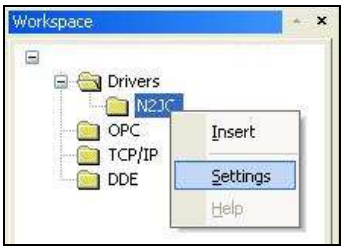
Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header**, and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header**,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

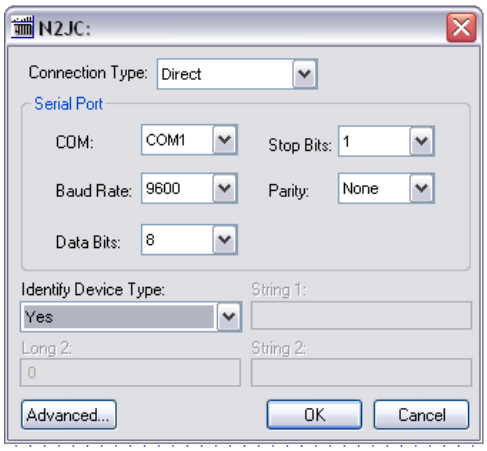
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system:

1. From the Studio development environment, select the *Comm* tab located below the *Workspace*.
2. Click on the *Drivers* folder in the *Workspace* to expand the folder.
3. Right-click on the N2JC subfolder and when the pop-up menu displays (as shown in the following figure), select the **Settings** option.



Select Settings from the Pop-Up Menu

The *N2JC: Communication Parameters* dialog displays (as follows).



Communication Parameters Dialog

Parameters	Default Values	Valid Values	Description
Identify Device Type	Yes	Yes or No	When this parameter is set to Yes, the driver requests the N2 device to respond with a code identifying that it is a valid device. Some devices do not accept this command, in this case this parameter must be set to No.

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, the driver and the host. You must be familiar with the DCE specifications before adjusting these configuration parameters.
> - The device must be configured with *exactly the same* parameters that you configured in the *N2JC Communication Parameters* dialog.

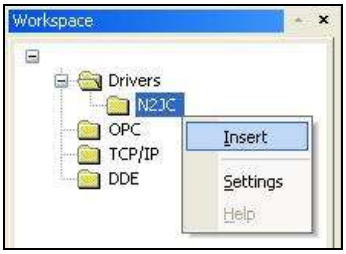
Configuring the Driver Worksheets

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*:

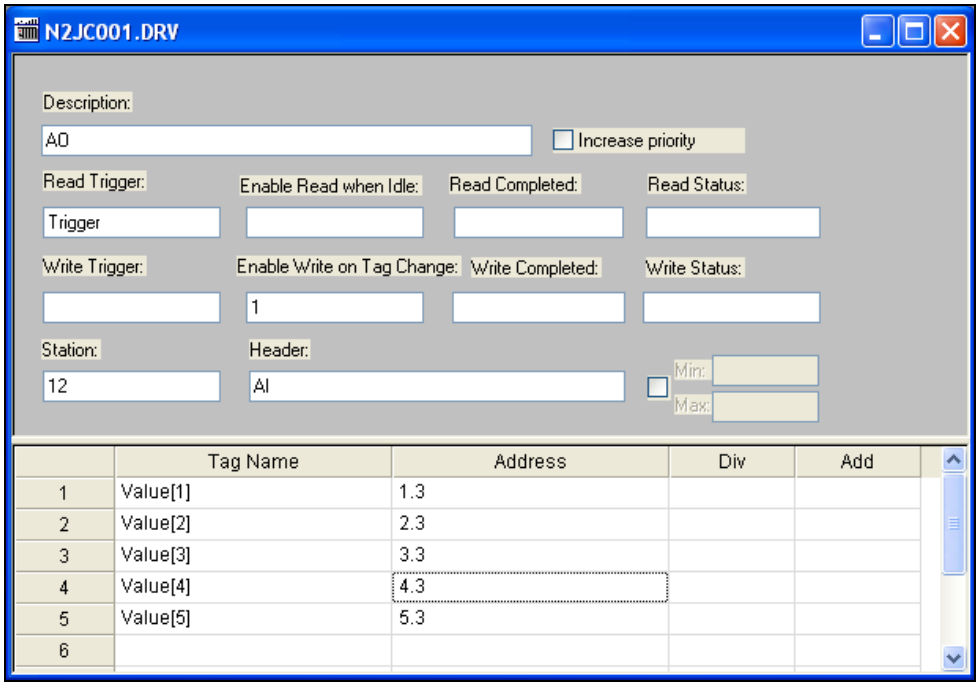
1. From the Studio development environment, select the *Comm* tab, located below the *Workspace* pane.
2. In the *Workspace* pane, expand the *Drivers* folder and right-click the <*Driver Name*> subfolder.
3. When the pop-up menu displays (as shown in the following figure), select the **Insert** option.



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: Specify the device using the following syntax:

  <Address>

  Where:
  – **Address** is device´s address.

- **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 and a reference to the initial address.

These variables must comply with the following syntax:

<Type> (For example: AI)

Where:

– **Type** is the object type. (AI, BI, AO, BO, BD, ADI and ADF)

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 N2JC driver.

Data Types	Comments
AI	Analog Input object
BI	Binary Input object
AO	Analog Output object
BO	Binary Output object
BD	Binary Data object
ADI	Analog Data Integer object
ADF	Analog Data Float object

- **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:

<Object>.<Attribute> (For example: 6.3)

Where:

– **Object** is the object number into the specified type.

– **Attribute** is the attribute number into the specified object.

Type	Attribute	Description
AI (Analog Input)	1 (1 Byte)	Object Configuration Bit 0 – COS_enabled (1) Bit 1 – unused Bit 2 – unused Bit 3 – alarm enabled (1) Bit 4 – warning enabled (1) Bit 5 – unused Bit 6 – unused Bit 7 – unused
	2 (1 Byte)	Object Status Bit 0 – Reliable (0) Bit 1 – Override active (1) Bit 2 – Out of range – high (1) Bit 3 – Out of range – low (1) Bit 4 – COS status Bit 5 – COS status Bit 6 – COS status Where Bits 4, 5, 6 are: 000 – normal 001 – trouble (JCI) 010 – not available 011 – low warning 100 – low alarm 101 – high warning 110 – high alarm Bit 7 – unused
	3 (1 Float)	Analog Input Value
	4 (1 Float)	Linear Ranging Parameter 1 (JCI)
	5 (1 Float)	Linear Ranging Parameter 2 (JCI)
	6 (1 Float)	Linear Ranging Parameter 3 (JCI)
	7 (1 Float)	Linear Ranging Parameter 4 (JCI)
	8 (1 Float)	Low Alarm Limit
	9 (1 Float)	Low Warning Limit
	10 (1 Float)	High Warning Limit
	11 (1 Float)	High Alarm Limit
	12 (1 Float)	Differential
	13 (1 Float)	Filter Weight (JCI)
	14 (1 Float)	AI_Offset (JCI)
BI (Binary Input)	1 (1 Byte)	Object Configuration Bit 0 – COS_enabled (1) Bit 1 – normal state Bit 2 – unused Bit 3 – alarm enabled (1) Bit 7 – unused
	2 (1 Byte)	Object Status Bit 0 – Reliable (0)

Type	Attribute	Description
		Bit 1 – Override active (1) Bit 2 – unused Bit 3 – unused Bit 4 – Alarm (1) Bit 5 – Trouble (1) (JCI) Bit 6 – current state Bit 7 – unused
	3 (Integer)	Debouncing Value in Msec (1-65535) (JCI)
	4 (Integer32)	Accumulator value (JCI)
AO (Analog Output)	1 (1 Byte)	Object Configuration Bit 0 – COS_enabled (1) Bit 1 – unused Bit 7 – unused
	2 (1 Byte)	Object Status Bit 0 – Reliable (0) Bit 1 – Override active (1) Bit 2 – Saturated high (1) (JCI) Bit 3 – Saturated low (1) (JCI) Bit 4 – unused Bit 7 – unused
	3 (1 Float)	Current Value
	4 (1 Float)	Low Linear Ranging Parameter (JCI)
	5 (1 Float)	High Linear Ranging Parameter (JCI)

Type	Attribute	Description
BO (Binary Output)	1 (1 Byte)	Object Configuration Bit 0 – COS_enabled (1) Bit 1 – normal state Bit 2 – unused
	2 (1 Byte)	Object Status Bit 0 – Reliable (0) Bit 1 – Override active (1) Bit 2 – unused Bit 3 – unused Bit 4 – Alarm (1) (JCI) Bit 5 – Trouble (1) (JCI) Bit 6 – current state Bit 7 – unused
	3 (Integer)	Minimum On-time (sec) (0-65535)
	4 (Integer)	Minimum Off-time (sec) (0-65535)
	5 (Integer)	Maximum Cycles/Hour
	6 (Integer)	Interstage on delay (sec) (0-65535) (JCI)
	7 (Integer)	Interstage off delay (sec) (0-65535) (JCI)
BD (Binary Data)	1 (1 Byte)	Object Configuration Bit 0 – Reliable (0) Bit 1 – Override active (1) Bit 2 – unused … Bit 7 – unused
	2 (1 Byte)	Current Value
ADI (Analog Data Integer)	1 (1 Byte)	Object Configuration Bit 0 – Reliable (0) Bit 1 – Override active (1) Bit 2 – unused … Bit 7 – unused
	2 (Integer)	Current Value
ADF (Analog Data Float)	1 (1 Byte)	Object Configuration Bit 0 – Reliable (0) Bit 1 – Override active (1) Bit 2 – unused … Bit 7 – unused
	2 (Float)	Current Value

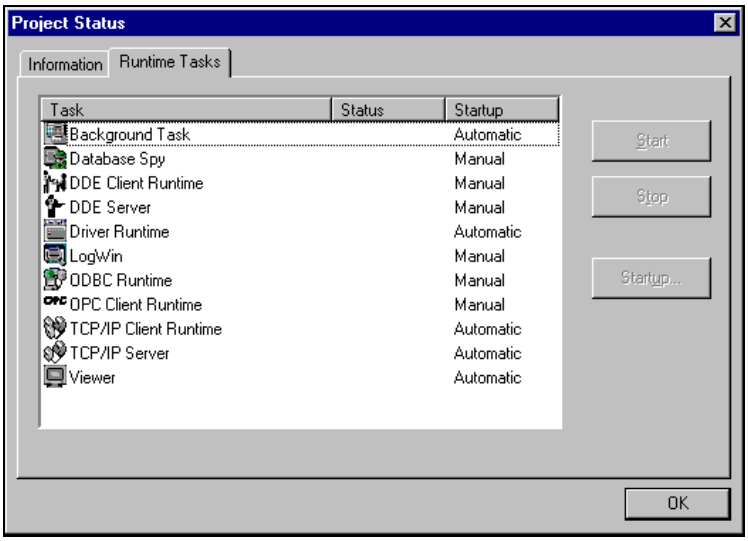
Executing the Driver

After adding the N2JC 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 box displays, as follows.



Project Status Dialog Box

2. Verify that the *Driver Runtime* task is set to **Automatic**.
   - If the setting is correct, click **OK** to close the dialog box.
   - 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 N2JC 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.
1	ERROR PROTOCOL	The driver received an invalid message.	Check the Type, Object and Attribute configured in driver.
2	ERROR INVALID COMMAND	Write command is invalid to the Type specified on the Header field.	Check if the Type specified on the Header field accepts write command.
3	ERROR INVALID ATTRIBUTE	Attribute configured on the Address field is invalid to the Type specified on the Header field.	Check the valid attribute.
4	ERROR INVALID MSG	The driver received an invalid message.	Check the Type, Object and Attribute configured in driver.
5	ERROR_IDENTTYPEDEV	Error when identify type of device	Some Devices do not accept this function, set Identify Device Type parameter to No. Check the Type, Object and Attribute configured in driver. 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.
-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 Box.
- **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.

Sample Application

You will find a sample application for drivers in the **/COMMUNICATION EXAMPLES/N2JC** directory. We strongly recommend that you check if there is a sample application for this driver and use it to test the driver before configuring your own customized application, for the following reasons:

- To better understand the information provided in each section of this document.
- To verify that your configuration is working satisfactorily.
- To certify that the hardware used in the test (device, adapter, cable, and PC) is working satisfactorily before you start configuring your own, customized applications.

> **Note:**
> This application sample is not available for all drivers.

Use the following procedure to perform the test:

1. Configure the device's communication parameters using the manufacturer's documentation.
2. Open and execute the sample application.

> **Tip:**
> You can use the sample application screen as the maintenance screen for your custom applications.

Revision History

Doc. Revision	Driver Version	Author	Date	Description of changes
A	1.00	Eric Vigiani	June 16, 2004	Initial version
B	1.01	Fabio H.Y. Komura	October 29, 2004	- Fixed problem when communicating with 2 or more devices. - Fixed problem with WriteItem. Wrong frame.
C	1.02	Rafael R. Fernandes	Jan 7, 2008	- Implemented Internal Operands (Byte, Int and Float) - Included Identify Device Type option in communication parameters - Fixed Analog operands addressing - Operand's names changed. - Driver Ported for Windows CE.
C	1.03	Lourenço Teodoro	Dec 16, 2008	- Updated driver version, no changes in the contents.