

**Communication Driver DEVNM**

Driver DeviceNet Master for Hilscher or Synergetic Boards

**Index**

**1 INTRODUCTION..... 2**

**2 GENERAL CHARACTERISTICS ..... 3**

2.1 DEVICE CHARACTERISTICS ..... 3

2.2 LINK CHARACTERISTICS..... 3

2.3 DRIVER CHARACTERISTICS ..... 3

2.4 INFORMATION ABOUT CONFORMANCE TESTING ..... 4

**3 INSTALLATION ..... 5**

3.1 INSTALLING THE DRIVER ..... 5

3.2 OTHER SOFTWARE REQUIREMENTS ..... 6

**4 DRIVER CONFIGURATION ..... 6**

4.1 SETTINGS - COMMUNICATION PARAMETERS..... 6

4.2 DRIVER WORKSHEET ..... 7

4.3 STATION AND HEADER CONFIGURATION..... 8

4.4 ADDRESS CONFIGURATION ..... 9

*To Input and Outputs:* ..... 9

*To COMMSTATUS:* ..... 9

4.5 MAIN DRIVER SHEET (MDS) ..... 10

4.6 DEVICE CONFIGURATION ..... 11

*Master* ..... 11

*Slave* ..... 11

**5 EXECUTION ..... 12**

**6 TROUBLESHOOTING..... 13**

**7 APPLICATION SAMPLE ..... 15**

**8 HISTORY OF VERSIONS..... 16**

# 1 Introduction

The DEVNM driver enables communication between Studio system configured as a DeviceNet Master and other DeviceNet devices by the Hilscher board interface, 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

- **Compatible Equipment:**
  - Any device compliant with the DeviceNet protocol according with the Hilscher board specification;

↳ **Tip:** Refers to section 2.4 to see the Equipment used in the standard conformance tests for this driver.

### 2.2 Link Characteristics

- **Network board Manufacturer:**
  - Hilscher / Synergetic
- **Network board Model:** DeviceNet Master Hischer board:
  - CIF 30-DNM
  - CIF 104-DNM
- **Network board software:**
  - Software to configure the board: SyCon Configurator
  - Software to test the communication with the board: Synergetic CIFTTest

### 2.3 Driver Characteristics

- **Operating System:**
  - Windows 9x
  - Windows 2000
  - Windows NT
  - Windows CE x86 **only**


↳ **Tip:** 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:

- **DEVNM.INI:** Internal file of the driver, it should not be modified by the user.
- **DEVNM.MSG:** Error messages for each error code. It should not be modified.
- **DEVNM.PDF:** Provides detailed documentation about the driver.
- **DEVNM.DLL:** Compiled driver.

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

- **CIF Device Driver:** Hilscher board libraries.

 **Note:** The **CIF Device Driver** should be provided with the board. The Studio's DEVNM driver requires the libraries installed with the CIF Device Driver to run properly. The API requested by the DEVNM driver is **CIF32DLL.DLL** (for WinNT/2000 OS), **CIFCEDLL.DLL** (for Windows CE OS). These files are component of the CIF Device Driver.

## **2.4 Information about conformance testing**

- Master: x86 HMI.
- Slave: Weg frequency inverter CFW-09 model
  
- **Configuration:**

SyCon configuration project: InduTest.pb

Baud Rate: 125 k

Protocol: DeviceNet

Master Hilscher / Synergetic Board Characteristics

Model: COM-DNM

TYPE: CIF104-BSL-DNM

- Operating System (development): Windows NT 4.0 + Service pack 4, Windows 9x
- Operating System (target): Windows CE v3.00
- Studio Version: 4.4
- Driver version: 1.00

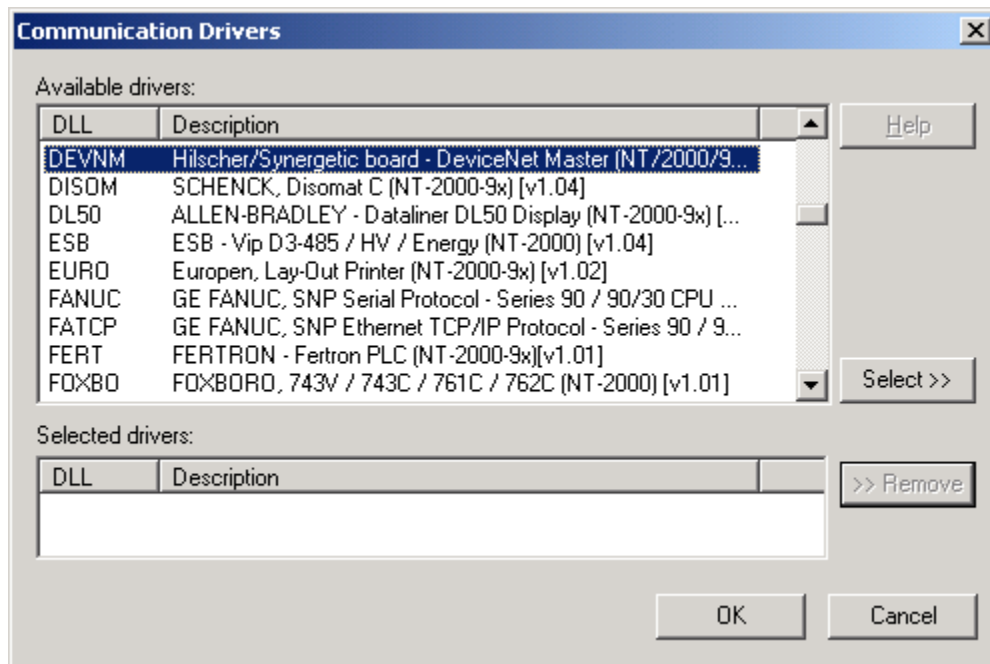
## 3 Installation

### 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 **DEVNM Driver** and push the button **Select >>** (the driver DEVNM must appear in the column **Selected Drivers**).
4. Press **OK**.



### 3.2 Other software requirements

- Windows NT/2000:

The Synergetic SyCon and CIFTTest are required to configure and test the board. Please refer to the Synergetic Documentation to know about how to use these software. Once configured by the SyCon and tested with the CIFTTest the Studio Driver will be able run successfully.

- Windows CE:

You must have the following Synergetic or Hilscher files in your CE Unit, compiled for you processor:

- CifCEdll.dll (windows folder or in the same folder where CEView is)
- CifISA.dll (windows folder)
- DrvSetup.exe
- CifTest.exe

These EXE files above will let you configure and test the board and then only the dlls are required to run the Studio DEVNM driver. Once the CifTest.EXE program runs with no errors, specially in the DevExchangeIO() function, the DEVNM driver will be able to run as well.

➤ **Attention:** Special care 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.

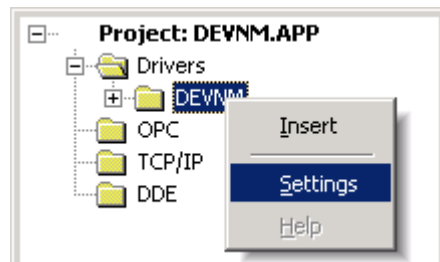
The driver configuration is two parts:

- The Settings or Communication parameters, it is only one configuration to the whole driver;
- The communication tables or Driver Worksheets, where the communication tags are defined. There are two types of communication tables: **Standard Tables** and **MAIN DRIVER SHEET**.

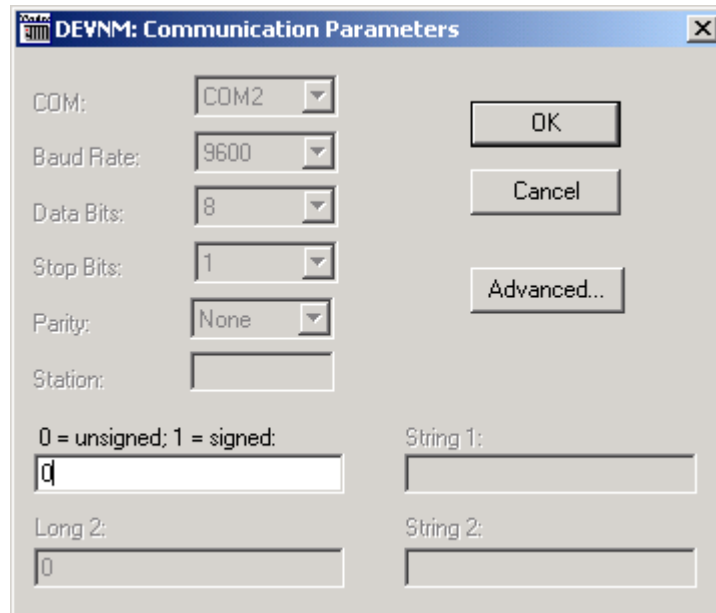
### 4.1 Settings - Communication Parameters

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 **DEVNM**.
3. Right click on the **DEVNM** subfolder and select the option **Settings**.



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



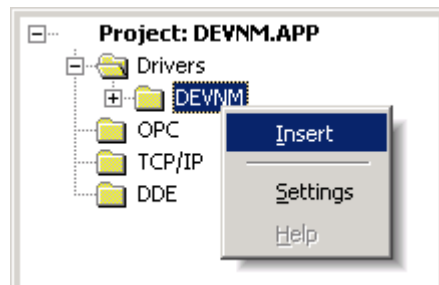
Parameter	Default Value	Valid values	Description
Station	0	0	Not used for the driver
0 = unsigned; 1 = signed	0	0 or 1	<ul style="list-style-type: none"> <li>▪ Unsigned:                             <ul style="list-style-type: none"> <li>- From 0 up to 255 to byte values;</li> <li>- From 0 up to 65535 to word values.</li> </ul> </li> <li>▪ Signed:                             <ul style="list-style-type: none"> <li>- From -128 up to 127 to byte values;</li> <li>- From -32768 up to 32767 to word values.</li> </ul> </li> </ul>

**Note:** All the other parameters (serial settings) are not used for this driver.

## 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 **DEVNM**.
3. Right click on the **DEVNM** subfolder and select the option **Insert**.



	Tag Name	Address	Div	
1	Word[1]	W0		
2	Word[2]	W2		
3	Byte[1]	B0		
4	Byte[2]	B1		
5	Byte[3]	B2		
6	Byte[4]	B3		
7	Word[4]	W4		

**Tip:** 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.

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

### 4.3 Station and Header configuration

Parameter	Default Value	Valid values	Description
Station	-	1 to 31, or 255	The Board's Address. It can also be used the value of 255, that means direct point to point communication with any station.
Header	0	See next table	Defines the type of instruction like, to read or write from or to the device from the reference of the initial address, or to get the Network Status and parameters.



The **Header** field defines the initial address both to the Inputs as to the outputs. It's possible to read the network status using the Header COMMSTATUS. Look forward to see these descriptions.

The parameter **Header** defines the type of variables that will be read or written from or to the host. It complies with the following syntax:

- To Input and Output:  
**<AddressReference>** (e.g.: 15);
- To Communication Status:  
**COMMSTATUS**
- To Communication Parameter:

- **AddressReference**: Initial Address (reference) of the memory address to be read/written.

Information regarding the parameter "Header"			
Type	Sample of syntax	Valid range of initial Address	Comment
Input / Output	0	0 to 511	You must have different worksheets to read and write to the device. If you write something, you are writing in the OutPut board image but you read from the Input Board image, this means that although you have the same numbers you have different Memory addresses.
Communication Status	COMMSTATUS	-	This header allows you get the communication status. Looking at the Address chapter you will be able to see what kind of status this command brings

#### 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.

- To Input and Outputs:  
**<Format><AddressOffset>.<Bit>**
- To COMMSTATUS:  
**<StatusAddress>**

- **Format**: W to treat the values as words or B to treat them as bytes.

- **AddressOffset**: This parameter is added to the AddressReference (configured in the **Header** field) to compose the address of the memory to be read/written.

- **Bit**: bit number (from 0 up to 15) from the word address. It's an optional parameter;

- **StatusAddress**: Address of the Status to be read from the Hilscher board;

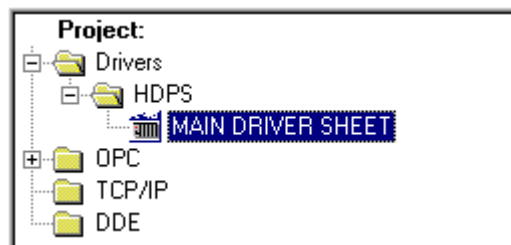
➤ **Attention:** If you are intending to communicate with Siemens devices you must have in mind these devices use to have an inversion of the byte order inside of a word. This driver only works in the HL (High-Low significance) byte order, but Siemens has the LH (Low byte as the most significant) byte order.

➤ **Attention:** It's NOT allowed to execute BIT writing. Only BIT reading is supported by this driver.

Sample of Addressing Configuration for Input and Output		
Address on the Master Device	Header Field	Address Field
IB 0	0	B0
IB10	0	B10
IB 10	10	B0
IB 10	5	B5
QB 1	0	B1
QB 217	200	B17
QW 0	0	W0
IW 100	50	W50

#### 4.5 Main Driver Sheet (MDS)

When the driver is inserted in the application, the MAIN DRIVER SHEET is automatically added to the driver folder.



The MAIN DRIVER SHEET provides a simple way to associate Studio tags to addresses in the PLC. Most of the MAIN DRIVER SHEET entries are standard for any driver. Refer to Studio Technical Reference Manual about the configuration of the standard fields. The fields which require specific syntax for this driver are described below:

	Tag Name	Station	I/O Address	Action	Scan	Div	Add
476	R_43129		W122	Read	Screen		
477	R_43130		W123	Read	Screen		
478	R_43201		W124	Read	Screen		
479	R_43202		W125	Read	Screen		

- **Station:** Not used.
- **I/O Address:** Address of each register from the PLC. The syntax used in this field is described below:
- To Input and Outputs:
  - **<Format><AddressOffset>.<Bit>** (e.g. W23.1)
    - **Format:** W to treat the values as words or B to treat them as bytes.
    - **AddressOffset:** This parameter is added to the AddressReference (configured in the **Header** field) to compose the address of the memory to be read/written.
    - **Bit:** bit number (from 0 up to 15) from the word address. It's an optional parameter.

⚠ **Attention:** It's NOT allowed to execute BIT writing. Only BIT reading is supported by this driver.

## 4.6 Device Configuration

### ▪ Master

To configure the Synergetic / Hilsher board you must use the SyCon software. Please look at the Synergetic/Hilsher SyCon software configuration to learn how to do that.

📌 **Tip:** You have to configure the network telling the Master all the Slaves it is going to communicate with, and then download the configuration to the board. To configure the slaves sometimes you need to get the Slave ESD file.

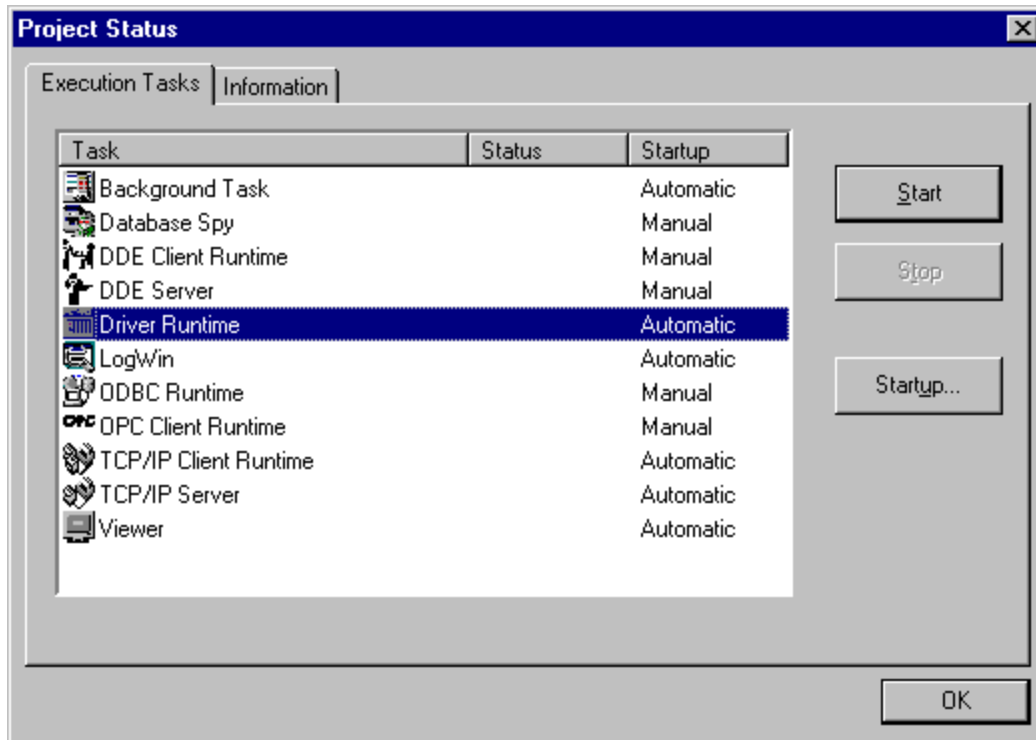
To test the board, you can both use the SyCon and the CIF Test program. In the Windows CE case maybe the board will need to be configured in a NT/95 station that has the SyCon software and then inserted on the CE box. The CIF Test for Windows CE from Synergetic is available by the Manufacture.

### ▪ Slave

The Slaves' configuration can be too much different depending on the manufacture. Please refer the Device's manufacture documentation to see their configuration instructions.

## 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




## 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	-
1	Invalid Station	The Station field contains a non-existing board address	Check on the driver configuration's worksheet that is getting this error if the configured station is right. If do not, correct it.
2	Invalid Header	An invalid Header has been typed or the tag that is inside this field has an invalid configuration.	Type a valid Header either on the header field or on the tag value. A lot of different valid headers are shown on the section 4.2
3	Invalid Address	An invalid address has been typed in the driver configuration worksheet	Check on the driver configuration's worksheet that is getting this error if the configured addresses are right. See this manual the valid addresses for each one of the valid Headers
4	Block size error	Address field greater then 512	The maximum address offset is 511. Correct it on the driver configuration worksheet
5	Protocol error	There is a protocol error	Run the COMMSTATUS function to check the protocol error
6	Checksum error	There is a protocol error	Run the COMMSTATUS function to check the protocol error
7	Error while opening the driver	Problems initializing the board	The board might not be configured, or connected. Run the SyCon, DrvSetup or The CifTets programs to detect the board configuration error
8	Error sending the message	There is a protocol or bus error	Run the COMMSTATUS function to check the protocol error
9	Error Receiving the message	There is a protocol or bus error	Run the COMMSTATUS function to check the protocol error
10	Invalid Offset	Address field greater then 512	The maximum address offset is 511. Correct it on the driver configuration worksheet
19	Reset Error	Error during the Reset Operation. It can be because of a conflict when using device interrupts or, as the timeout period can differ between fieldbus protocols, you must be experiencing timeout problems	Contact our Technical Support Team and describe the error
20	COMMSTATUS invalid address error	Address field out of the range between 1 and 13	Type a valid address, between 1 and 13,
21	COMMPARAM invalid address error	Address field out of the range between 0 and 1, or M1.TYPE (or M1.LENGTH) to M24.TYPE (or M24.LENGTH)	Type a valid address, between 0 and 1 or M1.TYPE (or M1.LENGTH) to M24.TYPE (or M24.LENGTH) in the COMMPARAM header case
25	Bit Address error in a word	Bit in the address field out of the range between 0 and 15	Type a valid address between Wx.0 and Wx.16
26	Bit Address error in a byte	Bit in the address field out of the range between 0 and 7	Type a valid address between Bx.0 and Bx.7

29	COMMSTATUS error	The operation (read, write) has returned an error.	Run the COMMSTATUS command and check the error
30	Invalid bit Operator error	Trying to write in a bit of a word or a byte	This driver is not able to write individual bits, only words and bytes
31	Error while exchanging messages	Error while reading an Input or writing an Output. These errors can be caused by: <ul style="list-style-type: none"> <li>- Timeout. The device needs more time then the defined by our driver</li> <li>- Wrong or no interrupt selected.</li> </ul>	<ul style="list-style-type: none"> <li>- Check interrupt on the device and in driver registration. They have to be the same! Interrupt already used by an other PC component.</li> <li>- Contact our technical support team</li> </ul>
32	Write parameter error	Wrong parameter in the COMMPARAM header writing case	Check the valid parameters to this operation. You can also consult the device manufacture documentation to see the right parameters' values.
-15	Timeout waiting start a message.	<ul style="list-style-type: none"> <li>- Disconnected cables</li> <li>- PLC turned off, or in Stop or error mode</li> <li>- Wrong Station number</li> <li>- Wrong RTS/CTS control settings.</li> </ul>	<ul style="list-style-type: none"> <li>- Check the cable wiring</li> <li>- Check the PLC state. It must be RUN</li> <li>- Check the station number.</li> <li>- Check the right configuration. See on the section 2.2 the different RTS/CTS valid configurations.</li> </ul>
-17	Timeout between rx char.	<ul style="list-style-type: none"> <li>- PLC in stop or error mode</li> <li>- Wrong station number</li> <li>- Wrong parity</li> <li>- Wrong RTS/CTS configuration settings</li> </ul>	<ul style="list-style-type: none"> <li>- Check the cable wiring</li> <li>- Check the PLC state. It must be RUN</li> <li>- Check the station number.</li> <li>- Check the right configuration. See on the section 2.2 the different RTS/CTS valid configurations.</li> </ul>

 **Tip:** The communication status can be verified by the **output** Window of the Studio's environment or by the **LogWin** module. 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, first of all establish the communication between the PLC Programming Tool and the PLC. Very frequently the communication 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 driver.

When testing the communication with the Studio, you should first use the application sample described at item 7 (if it's available), 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 documen

## 7 Application Sample

Studio provides 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.

 **Tip:** 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	Lourenco	10-Sep-2001	▪ First driver version