Communication Driver DEVN

Driver for communication with Synergetic's ISA card for DeviceNet Slave

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 5

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

5. EXECUTION 11

6. TROUBLESHOOTING 12

7. APPLICATION SAMPLE 14

8. HISTORY OF VERSIONS 15

1 Introduction

The DEVN driver enables communication between Studio system and Synergetic's ISA card for DeviceNet Slave using their protocol, 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:
  - Allen-Bradley

- Compatible Equipment:
  - Allen Bradley 500 Series with DeviceNet Module

- Allen-Bradley DeviceNet PLC programmer software:
  - DeviceNet Manager

☞ 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: DeviceNet port

- Physical protocol: DeviceNet

- Logic protocol: DeviceNet Protocol

- Device Runtime software: None

- Specific PC Board: SMS-ABDT-DNS DeviceNet AnyBus Slave Data Transfer

- Adapters / Converters: None

- Cable Wiring: DeviceNet Cable

2.3. Driver Characteristics

- Operating System:
  - Windows 9x
  - Windows 2000
  - Windows NT
  - Windows CE (Untested at this time)

☞ 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:

- Devn.INI: Internal file of the driver, it should not be modified by the user.
- Devn.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.

- Devn.PDF: This document provides detailed documentation about the driver.
- Devn.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: Synergetic ISA Card

Baud Rate: 250
Protocol: DeviceNet Slave

- Cable: According link specification, section 2.2.
- Operating System (development): Windows NT 4.0 + Service pack 4
- Operating System (target): Windows NT 4.0 + Service Pack 4
- Studio Version: 3.0 + SP4
- Driver version: 1.00

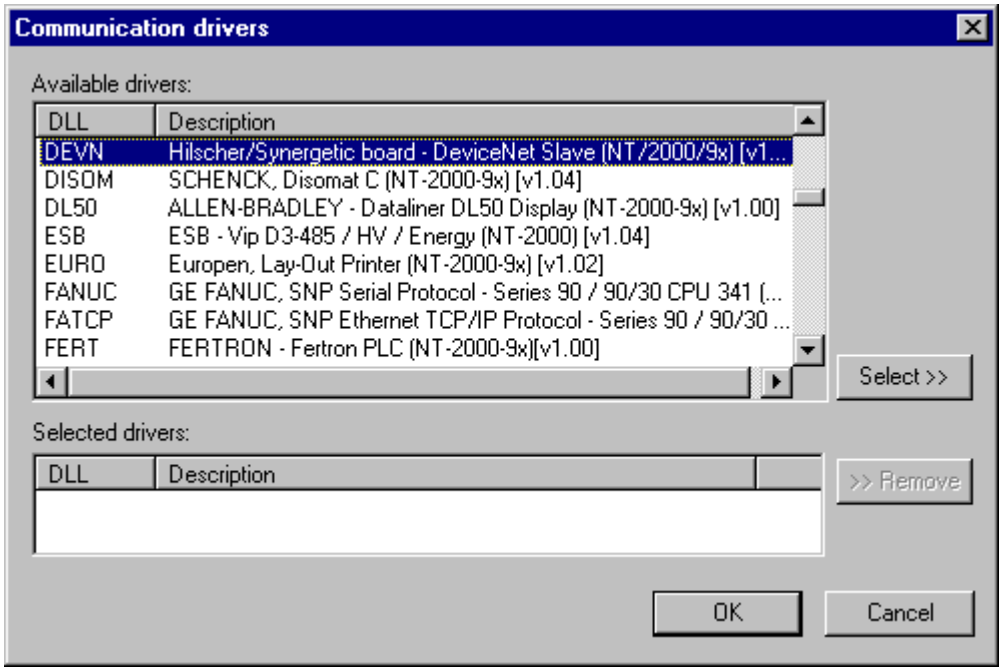
3. Installation

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



3.2. Other software requirements

It is necessary to install the following Dynamic Link Library (DLL) to enable the communication between Studio system and Hilscher's board:

- CIF32DLL.dll (Windows 9x/NT/2000)
- CIFCEDLL.dll (Windows CE)

Beside that, to download the custom program to the device, it is necessary to install one of PLC's programmer software, for example, DeviceNet Manager. Please see the DeviceNet Manager documentation about the procedure to install their software.

☞ Note: 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.

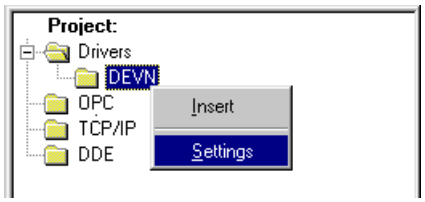
The driver configuration is two parts:

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

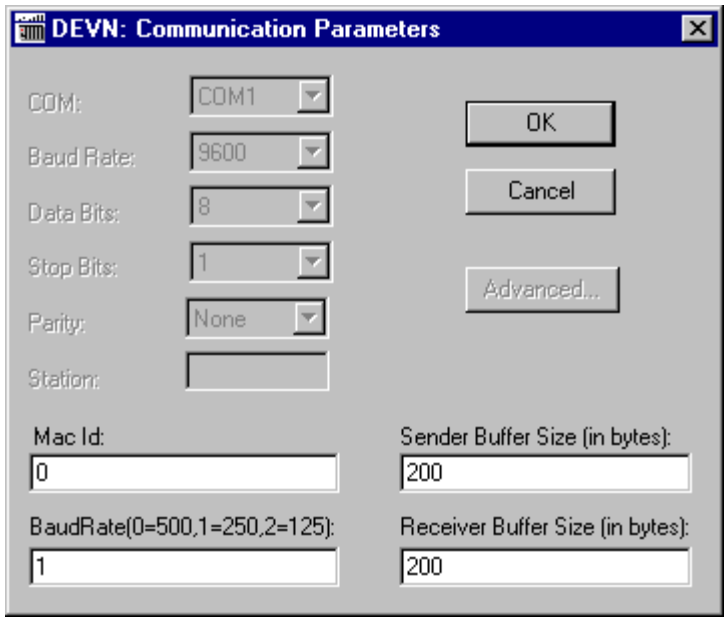
4.1. Settings - Communication Parameters

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



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



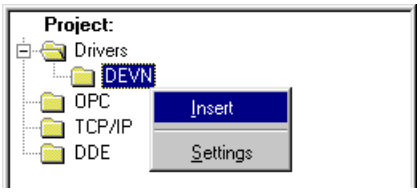
Parameter	Default Value	Valid values	Description
Mac ID	0	1 to 31	Must match with the node number assigned by host.
Baud Rate	1	0 to 2	Communication rate of data as set by host.
Sender Buffer Size (Bytes)	200	0 to 255	Host Buffer size.
Receiver Buffer Size (Bytes)	200	0 to 255	Node Buffer size.

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

4.2. Driver Worksheet

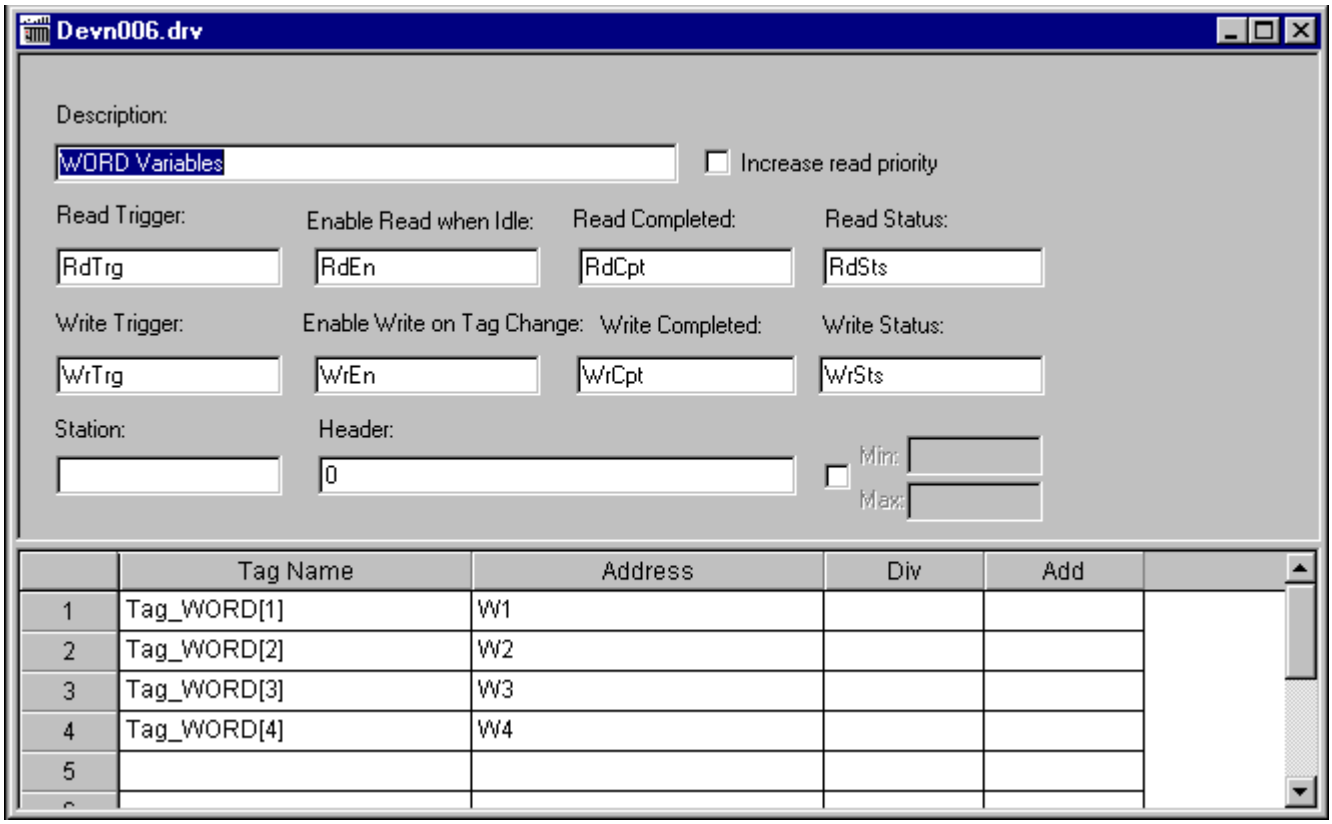
It is possible to configure many driver worksheets, each one will be composed of a Header and Body. To create a new driver worksheet, follow these steps:

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



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.

4.3. Station and Header configuration

Parameter	Default Value	Valid values	Description
Station	-	-	Not used. Only direct communication has been tested
Header	0	See next table	Defines the type of variable to be read or written from or to the device and the reference of the initial address.

The following table explain the valid values for the **Header** field:

Information regarding the parameter "Header"	
Header	**Comment**
<Initial Address>	Initial Address of the register to be read/write from/to device **in Words**. The final address is calculated adding this value to the offset specified in the **Address** field. Actual Destination Register and Valid Range is determined by the Host Setup. Where address 1 refers to the first register allocated to the node in question and the last valid address corresponds to the last register allocated to the Node.
COMMPARAM	Read/write the communication parameters from/to device.
COMMSTATUS	Read the communication status of the DeviceNet network.

☞ Note: For more information regarding the **Address** field configuration, refer section 4.4

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 syntax:

Information regarding the "Address" field		
Header Field	**Address Field**	**Comment**
<Initial Address>	<Type of register><offset>.<bit> or RESET.<type of Reset>	The **Address** field can be configured to read/write register in the device or reset the DeviceNet board installed in the system. To handle registers, this field must be configured with: • <type of register> : W for word, B for byte; • <offset> : this value will be added to the initial address to compose the final register's address in the device; • <bit> : the bit is optional and when used it specifies a bit, always from 0 to 7 for byte or 0 to 15 for word, in the register. The reset command (write only) must comply the following syntax: • RESET.1 = COLD RESET; • RESET.2 = WARM RESET; • RESET.3 = BOOT RESET
COMMSTATUS	<status register>	The status register (read only) can be filled with: • 1 = RX directly connected to CAN controller interrupt; • 2 = TX directly connected to CAN controller interrupt; • 3 = RxOverRun Incremented If the firmware does not read last message; • 4 = TxOverRun Incremented If the firmware does not transmit last message; • 5 = CAN chip error counter. If error, increments by 2, if good, decrements by 1 • 6 = should be different 0 • 7 = BusPrmValid • 8 = if CAN chip can send a message but received ACK does not comeback
COMMPARAM	<parameter register>	The parameter register can be: • 1 = Mac Id; • 2 = Baud Rate; • 3 = Produced buffer size in bytes; • 4 = Consumed buffer size in bytes;

To make effect the changes of the driver parameter, the user should send a RESET (cold or warm) command to the board.

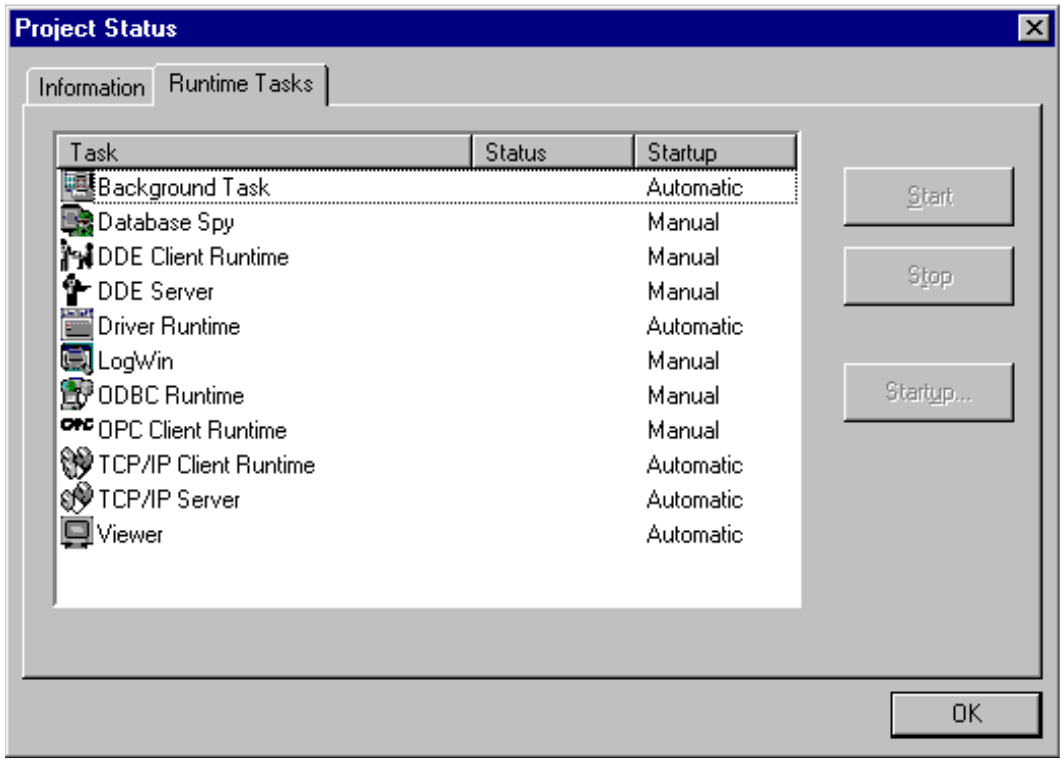
The following table shows some sample of the driver worksheet configuration.

Sample of Addressing Configuration		
Address on the Device/Command	**Header Field**	**Address Field**
W10	0	W10
W10	5	W5
B05	0	B5
B05	2	B1
W1.3 (bit 3 of the word register 1)	0	W1.3
W1.3 (bit 3 of the word register 1)	0	B2.3
WARM RESET	0	RESET.2
Mac Id register	COMMPARAM	1
CAN chip error counter	COMMSTATUS	5

☞ Note: In the previous table there are several ways to set the same variable on the device because of the variable's number defined by the sum of the **initial address reference** defined in the field header and the **offset** defined in the Address Field.
The Header addressing is always in words, but the address field setup can be in words or bytes depending on the type of variables used.
Important: if you configure an offset that reaches above the word register limit, it will be got invalid read values. For example, if you configure to read from W1 to W10, the read values will be inconsistent because you have only from W1 to W5. The read status still Ok, but the values are invalid.

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	-
2	Invalid Header Field	An invalid Header has been typed or the initial address is out-of-range (from 0 to 255).	Type a valid Header on the header field. A lot of different valid headers are shown on the section 4.3
3	Invalid Address Field	An invalid Address has been typed or the final address of the device register exceed the maximum value (255 bytes).	Type a valid Address on the address field. A lot of different valid address are shown on the section 4.4
4	Invalid register's address	An invalid register's address has been typed or the final address of the device register exceed the maximum value (255 bytes).	Make sure that the register's address doesn't exceed 255 bytes.
7	Error on initialize DeviceNet board	Board not properly configured, board is damaged.	Make sure that the board is properly configured using the vendor configuration software to check it.
9	Error on read device's registers	Board not configured properly, the network is crashed	Enable the LOGWIN to check the specific error returned by the command.
10	Invalid function	An invalid function was specified	Check: • Read/Write command in the Explicit Message worksheet; • Write command using COMMSTATUS
11	Invalid Mac Id	An invalid Mac Id was configured	Make sure that the Mac Id is from 0 to 63
12	Invalid Baud Rate	An invalid Baud Rate was configured	Check the baud rate of the network and the driver.
13	Invalid Consumed buffer size	The consumed buffer size is out-of-range	Valid values of the consumed buffer size are from 0 to 255 bytes
14	Invalid Produced buffer size	The produced buffer size is out-of-range	Valid values of the produced buffer size are from 0 to 255 bytes
15	Reset error	Error on try to reset the DeviceNet board	Enable the LOGWIN to check the specific error returned by the command.
17	Memory Error	The system couldn't allocate memory to store variables	Check the minimum memory requirements of the Studio and the integrity of them as well.
19	Invalid COMMSTATUS setup	Improper Address field using COMMSTATUS	Check if the Address field is properly configured to use COMMSTATUS
20	Invalid COMMPARAM setup	Improper Address field using COMMPARAM	Check if the Address field is properly configured to use COMMPARAM
21	Invalid RESET type	Invalid type of RESET command	Check the type of the RESET command. Valid values are from 1 to 3.
22	Invalid bit	An out-of-range bit was specified	Check the value of the bit. Valid values are from 0 to 15 for words or 0 to 7 for bytes
23	Invalid bit	An out-of-range bit was specified	Check the value of the bit. Valid values are

			from 0 to 15 for words or 0 to 7 for bytes
25	Invalid bit operation	Cannot perform bit operation	Check if the driver is using bit operation in: - COMMSTATUS, COMMPARAM (read/write); - Write registers operation and disable them.
26	Invalid initial address	Initial address exceed 255 bytes	Change the initial address to valid value
27	Invalid register's address	Register's address exceed 255 bytes	Change the register address to valid value. Remember that this value is the sum of initial address (header field) and the offset (address field)
28	Invalid Class	An out-of-range class was specified	Valid values for class are from 100 to 199
29	Invalid Instance	An out-of-range instance was specified	Valid values for instance are from 0 to 255
30	Invalid Attribute	An out-of-range attribute was specified	Valid values for attribute are from 0 to 255
31	Explicit Message TX error	Explicit Message TX Error	Enable the LOGWIN to check the specific error returned by the command.
32	Explicit Message RX error	Explicit Message RX Error	Enable the LOGWIN to check the specific error returned by the command.
34	Invalid Explicit Message Header	An explicit message request was received, but there's none driver worksheet associated with it	Check if there is a driver worksheet associated with the specific explicit message

☞ 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 A. Poon	8-May-2000	▪ First driver version