MODPL Communication Driver

Driver for Communication with Devices
Using Modbus Plus Protocol with PC Board

Contents

Introduction

The MODPL driver enables communication between the Studio system and devices using the Modbus Plus protocol communicating with PC Board, according to the specifications discussed in this document.

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

- **Introduction**: Provides an overview of the MODPL driver documentation
- **General Information**: Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the MODPL driver
- **Installing the Driver**: Explains how to install the MODPL driver
- **Configuring the Driver**: Explains how to configure the MODPL 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
- **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 XP/7 environment. If you are unfamiliar with Windows XP/7, 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 MODPL driver and the Modbus PLC.

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**: Modicon (or any device using the Modbus Plus protocol)
- **Compatible Equipment**:
  - AEG CPU 984 series
  - Schneider Modicon PLC Quantum CPU 213-04
  - Any device that is fully compatible with the Modbus Plus 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**: Modbus Plus Port
- **Physical Protocol**: RS-485
- **Logic Protocol**: Modbus Plus
- **Device Runtime Software**: None
- **Specific PC Board**: Schneider Modbus Plus boards, such as SA-85, PCI-85 or TSX C USB MBP

Driver Characteristics

The MODPL driver is composed of the following files:

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

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

- Windows XP

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

The MODPL driver supports the following registers:

Register Type	Length	Write	Read	Bit	Integer	Float
0x (Coil Status)	1 Bit	•	•	•	–	–
1x (Input Status)	2 Bit	–	•	•	–	–
3x (Input Register)	1 Word	–	•	•	•	–
4x (Holding Register)	1 Word	•	•	•	•	•

Conformance Testing

The following hardware/software was used for conformance testing:

- **Specific PC Board**: TSX C USB MBP
- **Cable**: Straight-through cable, Modbus Plus Cables
- **MBP Bridge** NW-BP85-002

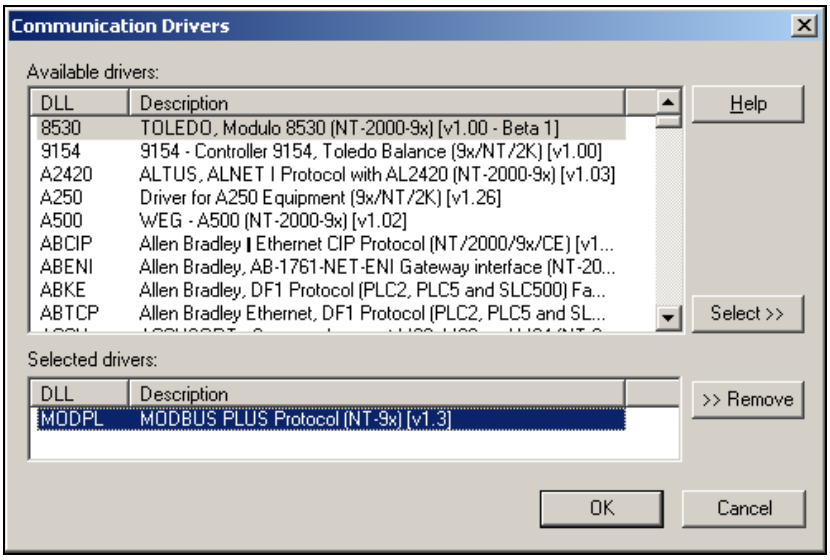
Driver Version	Studio Version	Operating System (development)	Operating System (runtime)	Equipment
1.11	7.1	Windows XP + Service Pack 3	Windows XP + Service Pack 3	2x Schneider Modicon PLC Quantum CPU 213-04 NW-BP85-002 MBP Bridge

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

> **Note:**
> This driver does not work without these DLL files (example: MBXAPI.DLL) that are installed with the MBP PC adapter.

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

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

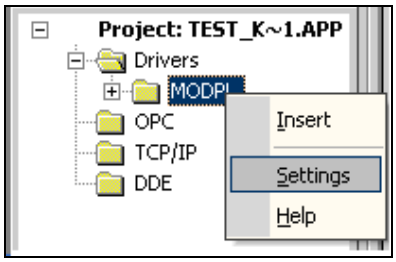
Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header**, and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header**,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

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

Setting the Communication Parameters

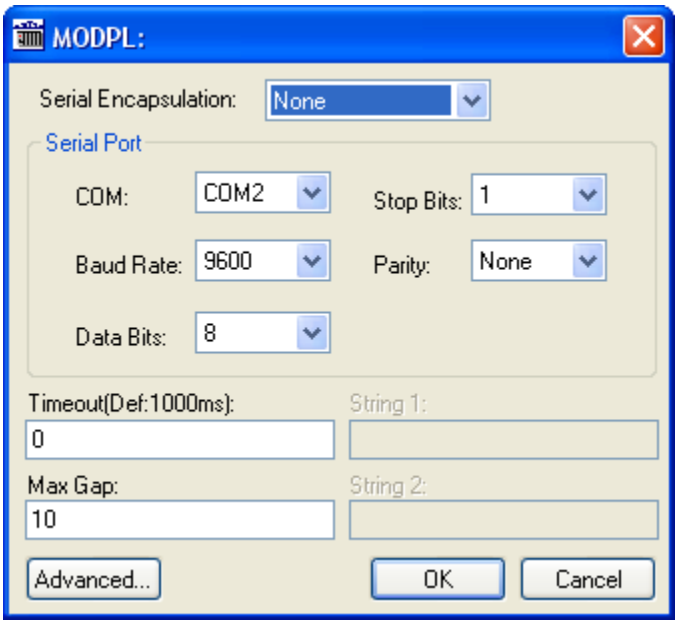
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system:

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 MODPL subfolder. When the pop-up menu displays (as shown in the following figure), select the **Settings** option.



Select Settings from the Pop-Up Menu

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



Communication Parameters Dialog

4. Specify the parameters as noted in the following table:

Parameters	Default Values	Valid Values	Description
Timeout(ms)	1000	0 to 32767	Timeout
Max Gap	10	0 to 1024	Indicates the maximum gap between consecutive addresses for virtual group creation. This gap is indicated in number of words, which means that every 16 consecutive addresses for Boolean headers (0x and 1x) count as 1 address for gap calculation. If this parameter is set to 0, the default value is used.

> **Note:**
> The device must be configured with *exactly the same* parameters that you configured in the *MODPL Communication Parameters* dialog.

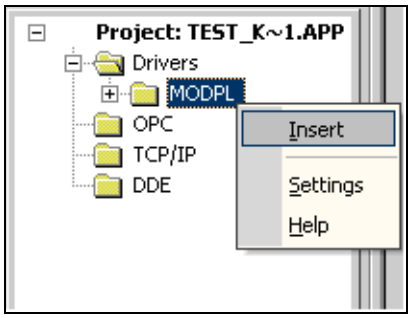
5. Click the **Advanced** button on the *Communication Parameters* dialog to open the *Advanced Settings* dialog and configure the settings as necessary.

> **Notes:**
> - Do not change any of the *Advanced* parameters at this time. You can consult the Studio *Technical Reference Manual* for information about configuring these parameters for future reference.

Configuring the STANDARD DRIVER SHEET

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

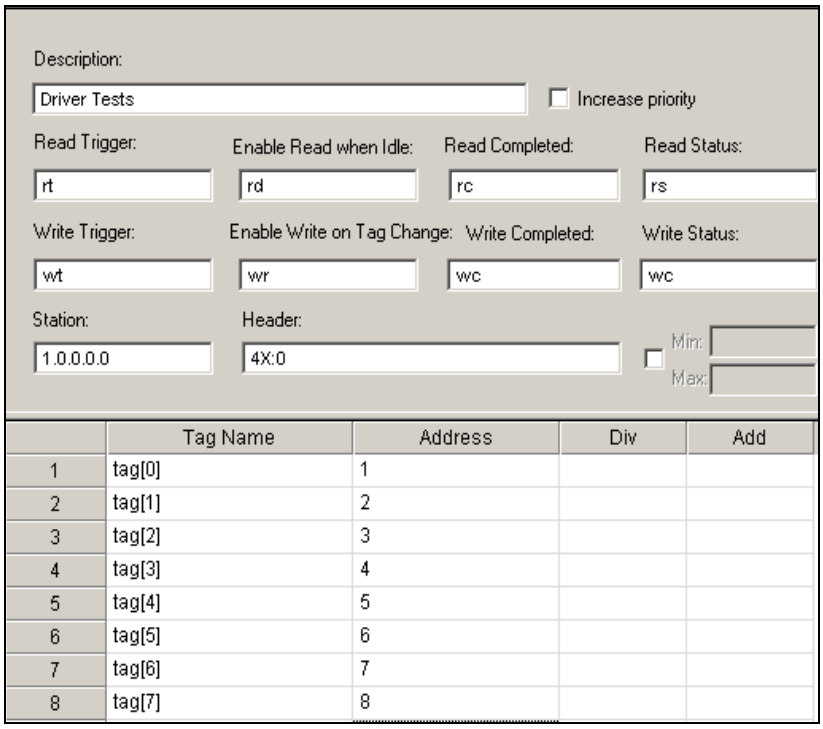
1. From the Studio development environment, select the *Comm* tab, located below the *Workspace* pane.
2. In the *Workspace* pane, expand the *Drivers* folder and right-click the *MODPL* subfolder.
3. When the pop-up menu displays (as shown in the following figure), select the **Insert** option.



Inserting a New Worksheet

> **Note:**
> To optimize communication, we recommend configuring the communication addresses in sequential blocks to improve performance.

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



	Tag Name	Address	Div	Add
1	tag[0]	1		
2	tag[1]	2		
3	tag[2]	3		
4	tag[3]	4		
5	tag[4]	5		
6	tag[5]	6		
7	tag[6]	7		
8	tag[7]	8		

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 for the Routing Path:

     <node number1>:<node number2>:<node number3>:<node number4>:<node number5>

     or

     <node number1>.<node number2>.<node number3>.<node number4>.<node number5>

     Where the first zero entry terminates the routing, the last non zero number is the node to be connected, and the other numbers are the routing.

     Example: **`22.24.3.0.0`** – This will connect with node number 3 through two Bridge Plus Devices, through address 22 on the local network, and then through address 24 on the second network.

> **Note:**
> The Modbus Plus network does not support Bridging devices for Global Data. In this case, use only the one single number to describe the node that you want to reach and communicate using Global Data

   - **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. The default value is **`4X:0`**.

     These variables must comply with the following syntax:

<*Type*>:<*AddressReference*> Example: **4X:10**

Where:

- **Type** is the register type. Valid values: **0X**, **1X**, **3X**, **4X**, **FP, GD** or **ID**
- **AddressReference** is the initial address (reference) of the configured type.

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. You must be certain that the tag's value is correct and that you are using the correct syntax; otherwise, you will get an invalid Header error.

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

Header Field Information			
Data Types	**Sample Syntax**	**Valid Range of Initial Addresses per Worksheet**	**Comments**
0X	0X:0	Varies according to the equipment	Coil status: Read and write events using Modbus instructions 01, 05, and 15
1X	1X:0	Varies according to the equipment	Input status: Read events using Modbus instruction 02
3X	3X:0	Varies according to the equipment	Input register: Read events using Modbus instruction 04
4X	4X:0	Varies according to the equipment	Holding register: Read and write events using Modbus instructions 03, 06, 16
FP	FP:0	Varies according to the equipment	Floating-point value (Holding register): Read and write float-point values using two consecutive Holding registers
GD	GD:0	1 to 32	Global Data: up to 32 Words of Global Data are possible per Modbus Plus node
ID	ID:0	Varies according to the equipment	Report Slave ID using Modbus instruction 17

- **Address** field: Use the information in the next table 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:

  <optFormat><*AddressOffset*>.[opt*Bit*] Example: **10, 20, DW40, 10.5**

  Where:

  - **optFormat** is a an optional parameter that can define the format of data for the headers **4X** and **GD.** The valid formats are: **S:** Signed Word, **DW**: Unsigned Double-Word, **SDW:** Signed Double Word, **FP**: Floating Point
  - **AddressOffset** is a parameter added to the **AddressReference** parameter (configured in the **Header** field), to compose the group address configured in the **Header** field.
  - **optBit** (*optional parameter used for GD (Global data) and 4X types [Holding register] only*) is the bit number to be read from or written to the device. It goes from 0 to 15.

Attention:

- The AddressReference plus AddressOffset result cannot equal zero (0). Modbus operands must start in an address that is greater than zero.
- Use Bit write commands are not possible
- The Double Words and Floating-point values are stored in two consecutive Holding registers or Global Data, where the address value corresponds to the first Holding register or Global Data position. You must not configure a non-existent address or a conflict will occur.

Note:

You may have noticed that there are 2 ways to access Floating Points on Holding Registers: you can either use the **`FP`** header or the **`4X`** header and configure the address format with **`FP`**

Address Configuration Sample		
Device Address	**Header Field**	**Address Field**
00001	0x:1	0
00010	0x:0	10
01020	0x:1000	20
10001	1x:1	0
10010	1x:0	10
11020	1x:1000	20
30001	3x:1	0
30010	3x:0	10
31020	3x:1000	20
40001	4x:1	0
40010 (signed)	4x:0	S10
41020	4x:1000	20
40010 (bit 0)	4x:0	10.0
41010 (bit 7)	4x:1000	10.7
40001 and 40002	FP:1	0
40013 and 40014 (Floating-point)	4x:0	FP13
41021 and 41022 (Floating-point)	FP:1000	21
41021 and 41022 (Floating-point)	4x:1000	FP21
40101 and 40102 (Double word)	4x:0	DW101
40201 and 40202 (Signed Double word)	4x:200	SDW1
GlobalData Word 10	GD:0	10
GlobalData Double Word 20	GD:0	DW20
GlobalData Word 12 Bit 7	GD:0	12.7

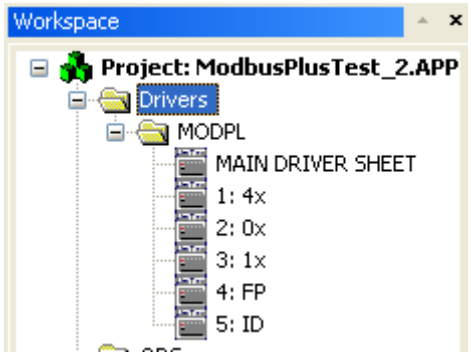
Attention:

You must not configure a range of addresses greater than the maximum block size (date buffer length) supported by each PLC within the same worksheet. The maximum data buffer length for this driver is 64 bytes per *STANDARD DRIVER SHEET*.

Configuring the MAIN DRIVER SHEET

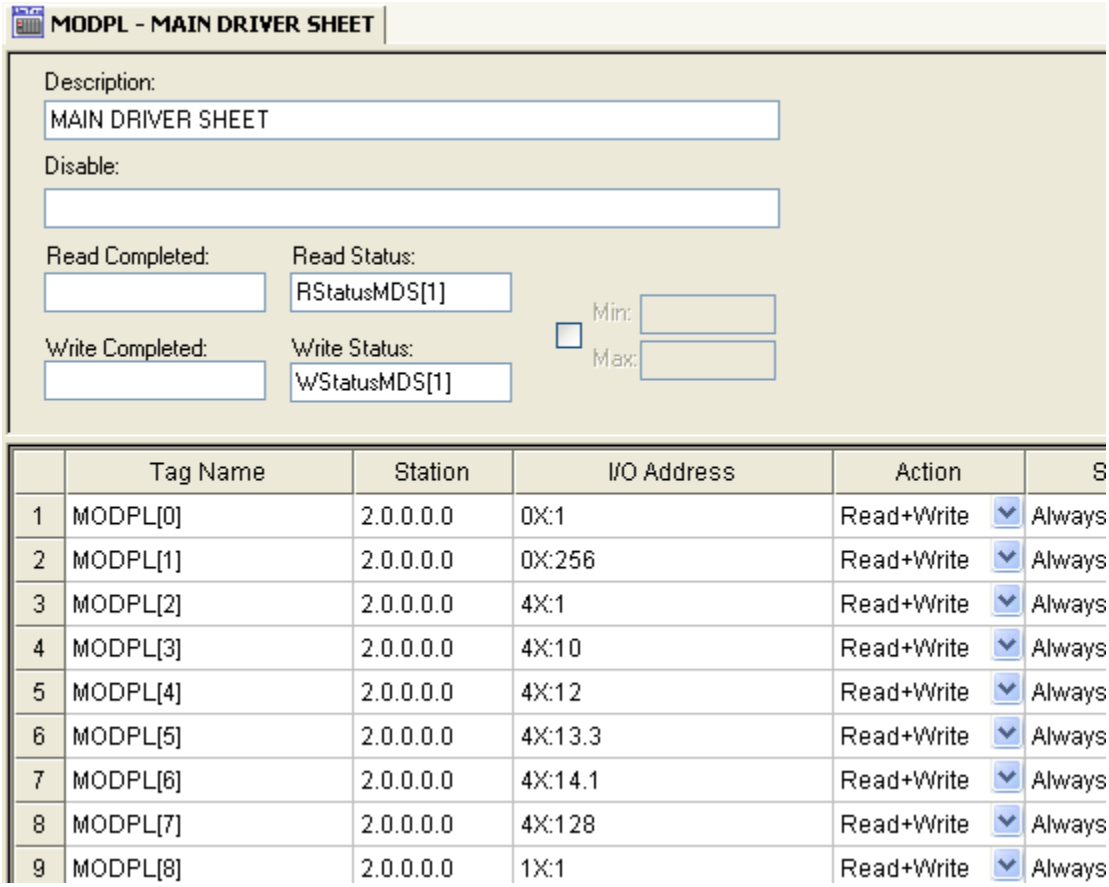
When you select the MODPL driver and add it to your application, Studio automatically inserts the Main Driver Sheet in the MODPL driver subfolder. To configure the Main Driver Sheet:

1. Select the Comm tab in the Workspace pane.
2. Open the Drivers folder, and then open the MODPL subfolder:



Main Driver Sheet in the MODPL Subfolder

3. Double-click on the MAIN DRIVER SHEET icon to open the following worksheet:



	Tag Name	Station	I/O Address	Action	S
1	MODPL[0]	2.0.0.0	0X:1	Read+Write	Always
2	MODPL[1]	2.0.0.0	0X:256	Read+Write	Always
3	MODPL[2]	2.0.0.0	4X:1	Read+Write	Always
4	MODPL[3]	2.0.0.0	4X:10	Read+Write	Always
5	MODPL[4]	2.0.0.0	4X:12	Read+Write	Always
6	MODPL[5]	2.0.0.0	4X:13.3	Read+Write	Always
7	MODPL[6]	2.0.0.0	4X:14.1	Read+Write	Always
8	MODPL[7]	2.0.0.0	4X:128	Read+Write	Always
9	MODPL[8]	2.0.0.0	1X:1	Read+Write	Always

Main Driver Sheet

Most of the fields on this sheet are standard for all drivers; see the "Communication" chapter of the Technical Reference Manual for more information on configuring these fields. However, the Station and I/O **Address** fields use syntax that is specific to the MODPL driver.

4. For each table row (i.e. each tag/register association), configure the **Station** and **I/O Address** fields as follows. Both fields were explained in detail on the previous section of this document. However, the **I/O Address** field demands a special treatment.

   - **I/O Address** field — On the *MAIN DRIVER SHEET* the address field contains both header and address of the *STANDARD DRIVER SHEET*, explained before. The following syntax shall be used:

     <Type>:<optFormat><Address>.[optBit]

     Where:

     – ***<Type>***: The types exposed before for the header field of the *STANDARD DRIVER SHEET*. It can be ***0x, 1x, 3x, 4x, FP, GD*** and ***ID.***

     – ***optFormat*** is a an optional parameter that can define the format of data for the headers 4X and GD. The valid formats are: **S:** Signed Word, **DW**: Unsigned Double-Word, **SDW:** Signed Double Word, **FP**: Floating Point

     – ***Address*** is the address number as it is on the PLC.

     – ***optBit*** (optional parameter used for GD (Global data) and 4X types [Holding register] only) is the bit number to be read from or written to the device. It goes from 0 to 15.

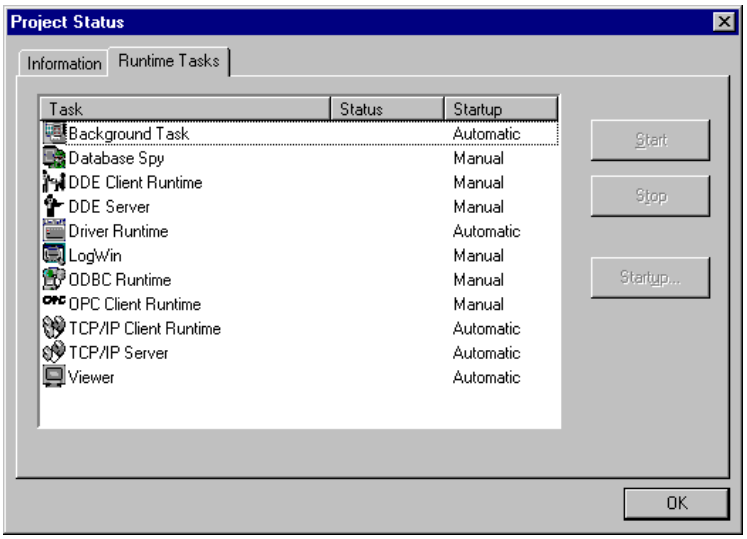
Executing the Driver

After adding the MODPL 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 MODPL 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 the failure that occurred.

Error Code	Description
0	OK
1	Illegal send or status buffer length
3	Invalid NetBIOS command
5	The timeout interval specified for a data send or receive command has expired.
6	Receive buffer too small
8	Invalid local session number in NCB_LSN
9	Out of resources
10	Session has been terminated by the remote node.
11	Command canceled
13	Duplicate local name
14	The local name table is full.
15	An ncb_close() command has been executed, but the name has active transactions.
17	The local session table is full.
18	Routing failure
19	Bad value in NCB_NUM field
20	No answer to CALL, or no such remote
21	Invalid name entry in local name table
22	Name already exists elsewhere on the network
23	Name has been incorrectly deleted
24	Session terminated abnormally-connection to remote node terminated
25	Name conflict-two nodes using the same name have been detected
26	Bad NetBIOS packet on network
27	Exceed limit of max nodes
28	Invalid Header (Ex: 0X, 1X, STA, 3X, 4X, ID, VP)
29	Invalid Address
30	Block length exceeds the limit.
32	Couldn't get node
33	Adapter busy-command cannot execute
34	Too many commands queued

Error Code	Description
35	Invalid value for NCB_LANA_NUM - must be 0 or 1
36	Command completed before ncb_cancel() executed
37	Modbus error
38	Invalid ncb_cancel() command - target NCB cannot be found
39	Bit write operation is not allowed
40	Invalid write operation – Read Only Memory
41	Addresses must be sequential to use the write trigger
64-254	Hardware errors
255	Still processing command

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 **Protocol Analyzer,** 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 remote Windows CE or XP Embedded target, you can use the Remote LogWin of Studio 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 **Help** → **Support 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

This driver does not have a sample application

Revision History

Doc. Revision	Driver Version	Author	Date	Description of changes
A	1.00	Sergio A. Poon	Dec/11/1997	Initial release
B	1.01	Sergio A. Poon	Oct/14/1999	Imported to UNICODE
C	1.02	Sergio A. Poon	Mar/20/2000	-Included code to read floats -Included code to allow more than n (default = 8) simultaneous connections
D	1.04	Leandro Coeli	Apr/22/2005	-Allowed Station like "1.0.0.0.0" -Implemented Auto-Complete on Header field
E	1.05	Leandro Coeli	May/25/2005	-Fixed problems on reading.
F	1.06	Leandro Coeli	May/31/2005	-Fixed problems on reading coils
G	1.07	Fabio Carvalho	Aug/08/2005	-Implemented Main Driver Sheet functions
H	1.08	Rafael R. Fernandes	Jan/19/2007	-Source Code Synchronization
I	1.09	André Körbes	Jun/04/2010	-Included Max Gap parameter and main driver sheet documentation. Updated troubleshooting section with new error codes (39, 40 and 41), and removed unused ones (12 and 31).
J	1.10	Paulo Balbino	Dec/02/2011	- Added Support to Global Data - Added support to signed and unsigned DWord and Signed values for words
K	1.11	Paulo Balbino	Jan/2013	-Fixed problem on groups generation for multiples stations