

MODBU Communication Driver

Driver for Serial Communication
with Devices Using the Modbus Protocol

Contents

INTRODUCTION 2

GENERAL INFORMATION..... 3

 DEVICE CHARACTERISTICS 3

 LINK CHARACTERISTICS 3

 DRIVER CHARACTERISTICS 4

 CONFORMANCE TESTING 4

INSTALLING THE DRIVER 5

CONFIGURING THE DEVICE 6

CONFIGURING THE DRIVER 6

 SETTING THE COMMUNICATION PARAMETERS 6

 CONFIGURING THE STANDARD DRIVER WORKSHEET 9

 CONFIGURING THE MAIN DRIVER WORKSHEET 12

EXECUTING THE DRIVER 14

TROUBLESHOOTING 15

SAMPLE APPLICATION 17

REVISION HISTORY..... 18

Introduction

The MODBU driver enables communication between the Studio system and a set of devices using the Modbus protocol by serial link, according to the specifications discussed in this document.

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

- **Introduction:** Provides an overview of the MODBU driver documentation.
- **General Information:** Provides information needed to identify all the required components (hardware and software) used to implement communication between Studio and the MODBU driver.
- **Installing the Driver:** Explains how to install the MODBU driver.
- **Configuring the Device:** Explains how to configure the Modbus device.
- **Configuring the Driver:** Explains how to configure the communication 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 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 product’s *Technical Reference Manual*.
- This document also assumes that you are familiar with the Windows NT/2000/XP environment. If you are unfamiliar with Windows NT/2000/XP, 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 serial communication between the Studio MODBU driver and devices using the Modbus protocol.

The information is organized into the following sections:

- Device Characteristics
- Link Characteristics
- Driver Characteristics

Device Characteristics

To establish serial communication, you must use devices with the following specifications:

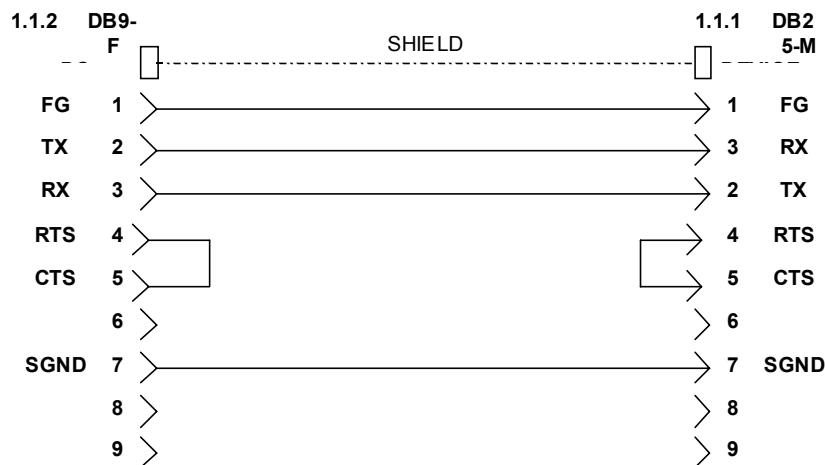
- **Manufacturer:** Modicon or any other device using the Modbus protocol for serial communication
- **Compatible Equipment:**
 - AEG CPU 984 Series
 - AEG CPU Compact
 - Siemens S7 S200 (using a special ladder program to emulate the Modbus protocol on the free port)
 - Any device that is compatible with the Modbus protocol
- **Modicon Programmer Software:** ModSoft

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

Link Characteristics

To establish serial communication, you must use links with the following specifications:

- **Device Communication Port:** Modbus Port (AEG 985E) and Free Port (S7-S200)
- **Physical Protocol:** RS232
- **Logic Protocol:** Modbus
- **Device Runtime Software:** None
- **Specific PC Board:** None
- **Cable Wiring Scheme:**



Cable Wiring Scheme

Driver Characteristics

The MODBU driver is composed of the following files:

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

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

- Windows 9x
- Windows 2000
- Windows NT
- Windows CE (x86, SH3, SH4, MIPS, ARM, PPC)

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

The MODBU driver supports the following registers:

Register Type	Length	Write	Read	Bit	Integer	Float	DWord	BCD	BCD DW	STRING
0x (Coil Status)	1 Bit	•	•	•	–	–	–	–	–	–
1x (Input Status)	1 Bit	–	•	•	–	–	–	–	–	–
3x (Input Register)	1 Byte	–	•	•	•	•	•	•	•	–
4x (Holding Register)	1 Byte	•	•	•	•	•	•	•	•	•

Conformance Testing

The following hardware/software was used for conformance testing:

- **Configuration:**
 - **Modbus Port:** 1
 - **Baud Rate:** 9600
 - **Protocol:** RTU
 - **Data Bits:** 8
 - **Stop Bits:** 1
 - **COM Port:** COM1
- **Cable:** Use specifications described in the “Link Characteristics” section.

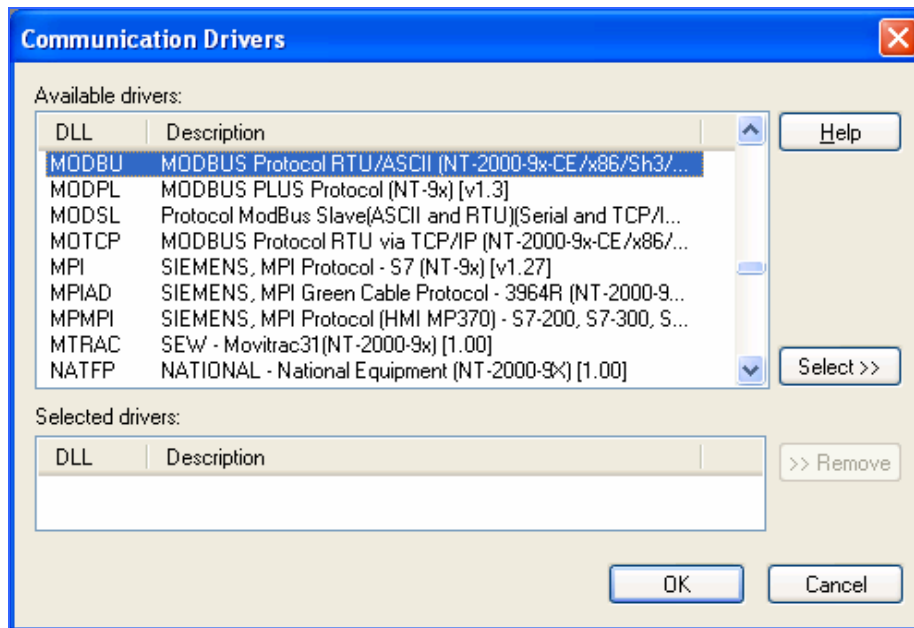
Driver Version	Studio Version	Operating System (development)	Operating System (target)	Equipment
2.14	6.0	WinXP	WinXP WinCEv4.00	PLC AEG CPU 984L

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 or double-click the Studio shortcut icon on your desktop.
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 **MODBU** driver from the *Available Drivers* list, and then click the **Select** button.



Communication Drivers Dialog

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

Note:

It is not necessary to install any other software on your computer to enable communication between Studio and the device. However, to download the custom program to your device, you must install a Modbus programmer software package (such as *ModSoft*). Consult the Modbus documentation for installation instructions.

Caution:

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 Device

Because there are several brands of equipment that use the Modbus protocol, it is impossible to define a standard device configuration. Therefore, we suggest using the following default configuration:

- **Protocol:** RTU
- **Baud Rate:** 9600
- **Data Bits:** 8
- **Stop Bits:** 1
- **Parity:** Even

Configuring the Driver

After opening Studio and selecting the MODBU driver, you must configure the driver. Configuring the MODBU driver is done in two parts:

- Specifying communication parameters (only one configuration needed).
- Defining communication tags and controls in the Communication tables or *Driver* worksheets (Standard and Main Driver Worksheets).

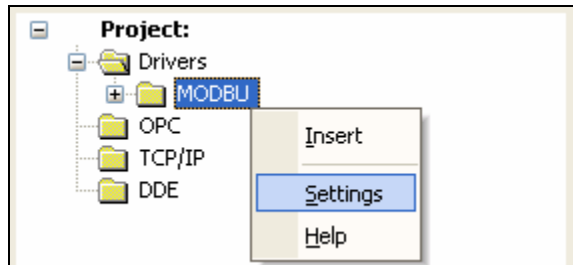
Worksheets are divided into two sections, a *Header* and a *Body*. The fields contained in these two sections are standard for all communications drivers — except the **Station**, **Header** and **Address** fields, which are driver-specific. This document explains how to configure the **Station**, **Header**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

Note:
For a detailed description of the Studio Standard and Main Driver Worksheets, and information about configuring the standard fields, review the product's *Technical Reference Manual*.

Setting the Communication Parameters

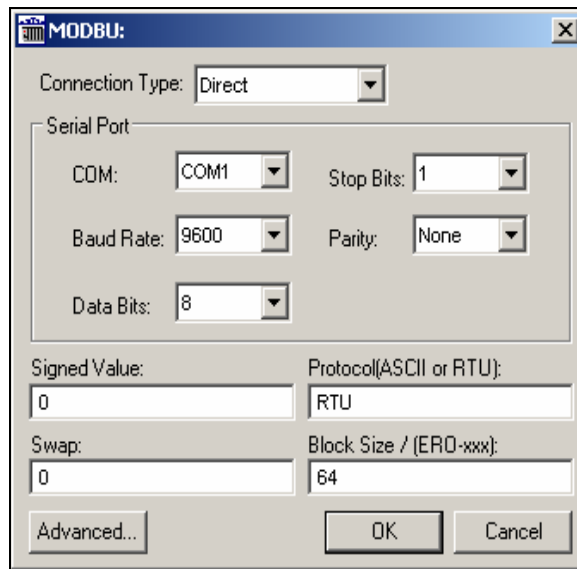
Use the following steps to configure the communication parameters, which are valid for all driver worksheets configured in the system):

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 *MODBU* subfolder and when the pop-up menu displays, select the **Settings** option:



Select Settings from the Pop-Up Menu

The MODBU: Communication Parameters dialog displays:



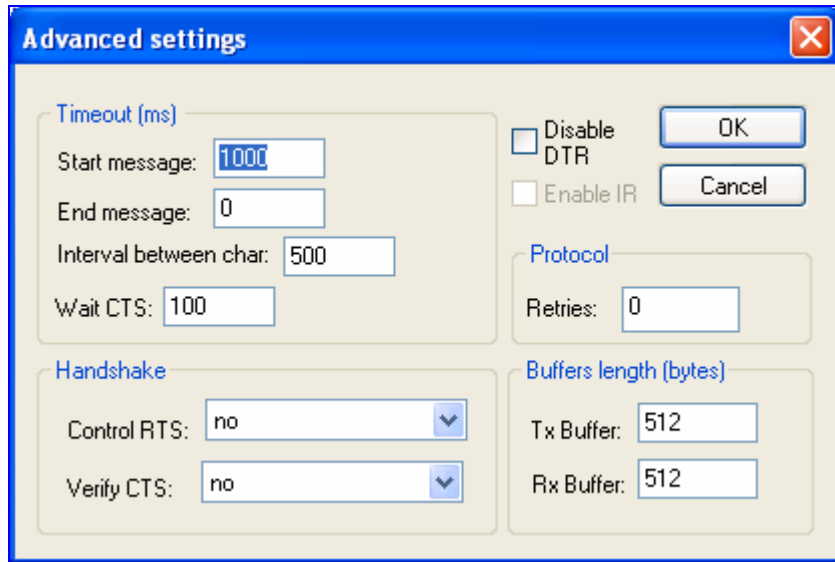
MODBU: Communication Parameters Dialog

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

Parameters	Default Values	Valid Values	Description
Station	0	0	Not required for this driver
0-Signed 1-Unsigned Value	0	0 or 1	0: Signed values 1: Unsigned values
Protocol (ASCII or RTU)	RTU	ASCII or RTU	Modbus protocol types
SwapWord (0=No/1=Yes)	0	0 or 1	Swap Words for FP, FPS, FP3, FP3S, DW, DWS, DW3 and DW3S register types
Block Size / (ERO-xxx)	—	0 – 512 or ERO-xxx	Block Size (Words) or Address used to set equipment to Local or Remote (used for ERO equipment <i>only</i>).

Note:
 These parameters must be configured *exactly the same* as those you configured for the MODBU driver in the *Communications Parameters* dialog.

5. Click the **Advanced** button on the *Communication Parameters* dialog to open the *Advanced Settings* dialog:



Advanced Settings Dialog

6. Use the following table to specify the **Control RTS** (*Request to Send*) parameter:

Parameter	Default Value	Valid Values	Description
Control RTS	No	no yes yes + echo	<p>Specify this parameter if the RTS handshake signal is set before communication and if there is an echo in the communication.</p> <ul style="list-style-type: none"> If you are using Windows 95 or CE with the correct RS232–RS485 Converter (without RTS Control), select no. If you are using Windows NT with the Cutler Hammer RS232–485 adapter, you must select yes. <p>Important: Setting this parameter incorrectly prevents the driver from working, and generates a Timeout waiting to start a message error message.</p>

Notes:

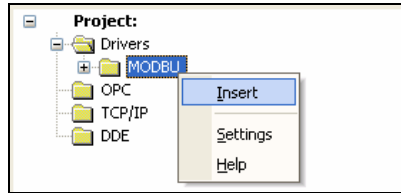
- Do not change any of the other default Advanced Settings parameters at this time. You can consult the *Studio Technical Reference Manual* for information about configuring these parameters for future reference.
- Generally, you must change these parameters only if you are using a DCE (Data Communication Equipment) converter (such as 232/485), modem, and so forth between your computer, the driver and the host. However, before adjusting the advanced communication parameters, you must be familiar with the characteristics of the DCE.

Configuring the Standard Driver Worksheet

This section explains how to configure a *Standard Driver Worksheet* (or Communication table) to associate application tags with the PLC addresses. You can configure multiple *Driver* worksheets — each of which is divided into a *Header* section and a *Body* section.

Use the following steps to create a new Standard Driver Worksheet:

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 *MODBU* subfolder.
3. When the pop-up menu displays, select the **Insert** option:

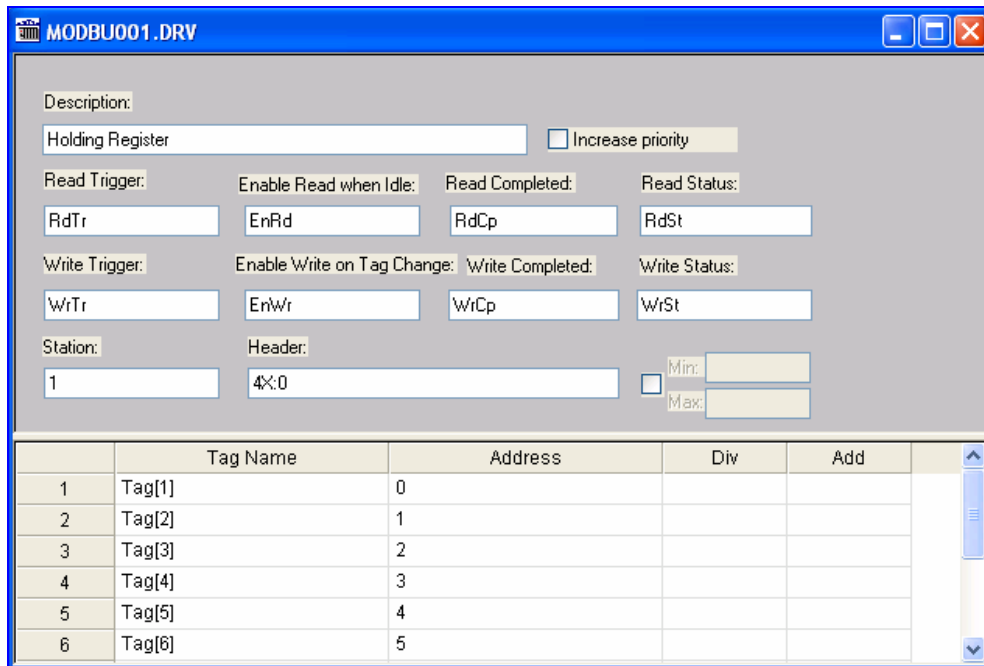


Inserting a New Worksheet

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 *MODBU.drv* dialog displays (similar to the following figure):



MODBU Driver Worksheet

4. Use the following information to complete the **Station**, **Header** and **Address** fields on this worksheet:
- **Station** field: Use this field to specify the ID (*node*) of the device (*unit name*). Valid values are 0–247 (*no default*).
 - **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 also a reference to the initial address. (Default value is 0X: 0.)
 - These variables must comply with the following syntax:
 <Type>: <AddressReference> (for example: 4X: 10)

Where:

- <Type> is the register type. Type one of the following: 0X, 1X, 3X, 4X, FP, FPS, FP3, FP3S, DW, DWS, DW3, DW3S, BCD, BCD3, BCDDW, BCDDWS, BCDDW3, BCDDW3S, ID or ST.
- <AddressReference> is the initial address (reference) of the register type you configured.

After you edit the **Header** parameter, the system checks that the syntax is valid. If the syntax is invalid, Studio automatically inserts the default value (0X: 0) into 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 **Header** field:

Header Field Information			
Data Types	Sample Syntax	Valid Range of Initial Addresses	Comments
0X	0X: 0	Varies according to equipment	Coil status: Reads and writes events using Modbus instructions 01, 05, and 15.
1X	1X: 0	Varies according to equipment	Input status: Reads events using Modbus instruction 02.
3X	3X: 0	Varies according to equipment	Input register: Reads events using Modbus instruction 04.
4X	4X: 0	Varies according to equipment	Holding register: Reads and writes events using Modbus instructions 03, 06 and 16.
FP	FP: 0	Varies according to equipment	Floating-point value (Holding Register): Reads and writes floating-point values using two consecutive Holding Registers.
FPS	FPS: 0	Varies according to equipment	Floating-point value (Holding Register): Reads and writes floating-point values using two consecutive Holding Registers with Byte Swap.
FP3	FP3: 0	Varies according to equipment	Floating-point value (Input Register): Reads floating-point values using two consecutive Input Registers.
FP3S	FP3S: 0	Varies according to equipment	Floating-point value (Input Register): Reads floating-point values using two consecutive Input Registers with Byte Swap.
DW	DW: 0	Varies according to equipment	DWord value (Holding Register): Reads and writes DWord values using two consecutive Holding Registers.
DWS	DWS: 0	Varies according to equipment	DWord value (Holding Register): Reads and writes DWord values using two consecutive Holding Registers with Byte Swap.
DW3	DW3: 0	Varies according to equipment	DWord value (Input Register): Reads DWord values using two consecutive Input Registers.
DW3S	DW3S: 0	Varies according to	DWord value (Input Register): Reads DWord values using two consecutive

Header Field Information			
Data Types	Sample Syntax	Valid Range of Initial Addresses	Comments
		equipment	Input Registers with Byte Swap.
BCD3	BCD3 : 0	Varies according to the equipment	BCD value (Input Register): Reads events using Modbus instruction 04.
BCD	BCD : 0	Varies according to the equipment	BCD value (Holding Register): Reads and writes events using Modbus instructions 03, 06 and 16.
BCDDW	BCDDW : 0	Varies according to the equipment	BCD 32-bit integer value (Holding Register): Reads and writes 32-bit integer values using two consecutive Holding Registers.
BCDDWS	BCDDWS : 0	Varies according to the equipment	BCD 32-bit integer value (Holding Register): Reads and writes 32-bit integer values using two consecutive Holding Registers with Byte Swap.
BCDDW3	BCDDW3 : 0	Varies according to the equipment	BCD 32-bit integer value (Input Register): Reads 32-bit integer values using two consecutive Input Registers.
BCDDW3S	BCDDW3S : 0	Varies according to the equipment	BCD 32-bit integer value (Input Register): Reads 32-bit integer values using two consecutive Input Registers with Byte Swap.
ID	ID : 0	Varies according to the equipment	Report Slave ID using Modbus instruction 17.
ST	ST : 0	Varies according to the equipment	String value (Holding Register): Reads and writes String values using consecutive Holding Registers.

- **Address field:** The body of the *Driver* worksheet allows you to associate each tag to its respective address in the device. 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:

[Signed/Unsigned]<AddressOffset> (for example, 10) or
 <AddressOffset>.<Bit> (for example: 10.3) or
 <AddressOffset>:<Len> (for example: 10:5) (Only for ST header) (Len is the length in **BYTES**)

Where:

- [Signed/Unsigned] (*optional parameter used for integer values only*): If you do not specify this parameter, Studio inserts an integer value based on the parameters you set in the *Communication Parameters* dialog. Valid values are **S** (signed) and **U** (unsigned).
- <AddressOffset> is a parameter added to the **AddressReference** parameter (configured in the **Header** field) to compose the address to be read from or written to the device.
- <Bit> is the bit number (from 0–15) of the word address. This parameter is *optional* and can be combined with 3X or 4X **Header** configuration.

⇒ **Cautions:**

- Studio stores the floating-point value in two consecutive words. In this case, the **Address** value corresponds to the first Holding Register position. You must not configure a non-existing address, or it will create a conflict.
- In the *Driver* worksheet, the start address is defined by the sum of the <AddressReference>

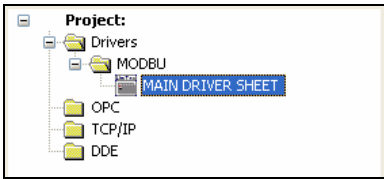
parameter (from the **Header** field) and the lowest **<AddressOffset>** parameter (in the **Address** field). The resulting start address *must be a non-zero value*. A zero value will cause a conflict.

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

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
31020 (bit 5)	3x:1000	20.5
40001	4x:1	0
40010	4x:0	10
41020	4x:1000	20
41020 (bit 0)	4x:1000	20.0
40001 and 40002	FP:1	0
40013 and 40014	FP:0	13
41021 and 41022	FPS:1000	21

Configuring the MAIN Driver Worksheet

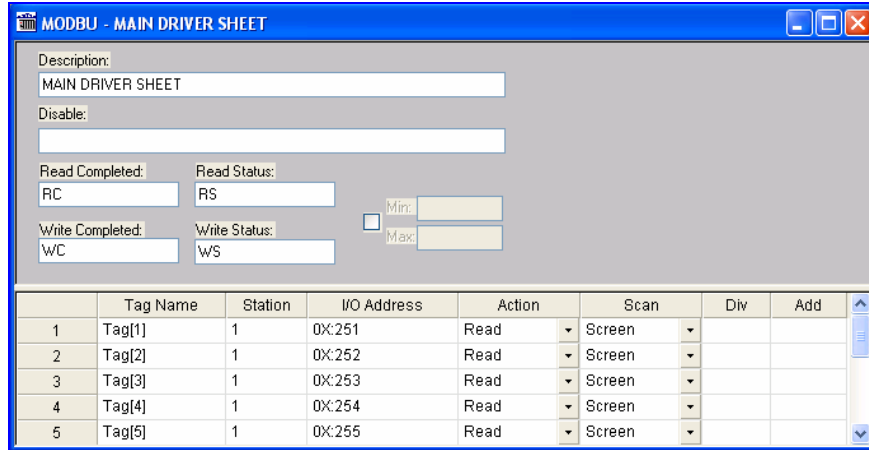
When you add the MODBU driver to your application, the program automatically adds the *MAIN Driver Worksheet (MDS)* to the *MODBU* driver folder as shown:



You can use the MDS to associate Studio tags to addresses in the PLC. Most MDS parameters are standard for any driver, and are not discussed in this publication. For information about configuring these parameters, consult the *Studio Technical Reference Manual*.

Use the following instructions to configure the parameters that are specific to the MODBU driver:

1. Double-click on the *Main Driver Sheet* icon to open the following worksheet:



2. Configure the following fields on this worksheet:

- **Station** field: Type the PLC address ID number
- **I/O Address** field: Type the address of each PLC register, using the following syntax:
 <Type>: [Signed/Unsigned]<Address> (for example, 4X:20) or
 <Type>:<Address>.<Bit> (for example, 4X:20.6)

Where:

- [Signed/Unsigned] (*optional*): Parameter used for integer values only. If you do not specify this parameter, Studio uses the *Communication Parameters* settings to configure integers. Valid values for this parameter are *s* (*Signed*) or *u* (*Unsigned*).
- <Type> is the register type. Type one of the following: 0X, 1X, 3X, 4X, FP, FPS, FP3, FP3S, DW, DWS, DW3, DW3S, BCD, BCD3, BCDDW, BCDDWS, BCDDW3, BCDDW3S or ID.
- <Address> is the register address of the device.
- <Bit> is the bit number (from 0–15) of the word address. This parameter is *optional* and can be combined with 3X- or 4X-type addresses only.

➔ **Caution:**
 You must use a non-zero value in the **Station** field, and you cannot leave the field blank.

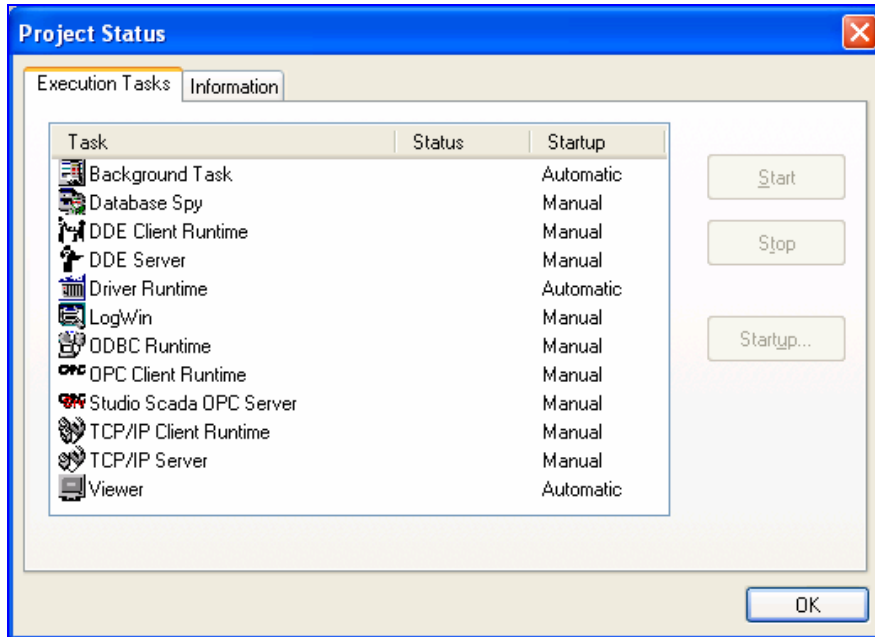
Executing the Driver

After adding the MODBU 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 displays:



Project Status Dialog

2. Verify that the *Driver Runtime* task is set to **Automatic**.
 - If the setting is correct, click **OK** to close the dialog.
 - 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 MODBU 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	Possible Causes	Procedure to Solve
0	OK	Communication without problems	None required
2	Invalid Address	Invalid address typed in Driver worksheet	Type a valid address.
3	Invalid data values	Specified address does not exist on the device so protocol received invalid data	Verify that specified address exists on the device.
4	Equipment failed	Equipment failed or out of order	Check equipment state.
5	Ack	Ack action error during communication	Check device and Studio Communication Parameters.
6	Equipment in use	Command invalid when equipment is in use	Studio commands cannot generate this error.
7	Negative Ack	Ack action error during communication	Check device and Studio Communication Parameters.
8	Memory parity error	Invalid Communication Parameter	Check driver Communication Parameters.
10	Invalid Header field	Specified invalid tag value in Header field	Specify a valid Header tag value.
11	Invalid Address field	Specified invalid Address	Specify a valid address.
12	Invalid block size	Offset greater than maximum allowed	Specify a valid offset or create a new worksheet. Typically, maximum offset is 64.
13	Invalid CRC	Invalid CRC in response message	<ul style="list-style-type: none"> ▪ Check the cable wiring. ▪ Check the station number. ▪ Check the RTS/CTS configuration (see <i>Studio Technical Reference Manual</i> for valid configurations).
18	Invalid BCD Value	Tried reading an invalid BCD value	Verify that PLC value is valid.
19	Invalid BCD Value	Tried writing a negative BCD value	Only positive BCD values are valid.
-15	Timeout Start Message	<ul style="list-style-type: none"> ▪ Disconnected cables ▪ PLC is turned off, in stop mode, or in error mode ▪ Wrong station number ▪ Wrong RTS/CTS control settings 	<ul style="list-style-type: none"> ▪ Check cable wiring. ▪ Check the PLC state – it must be RUN. ▪ Check the station number. ▪ Check the RTS/CTS configuration (see <i>Studio Technical Reference Manual</i> for valid configurations).
-17	Timeout between rx char	<ul style="list-style-type: none"> ▪ PLC in stop mode or in error mode ▪ Wrong station number ▪ Wrong parity ▪ Wrong RTS/CTS configuration settings 	<ul style="list-style-type: none"> ▪ Check cable wiring. ▪ Check the PLC state – it must be RUN. ▪ Check the station number. ▪ Check the RTS/CTS configuration (see “Link Characteristics” for 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 enable the log at the unit (**Tools** → **LogWin**) and verify the `ce1log.txt` file created at the target unit.

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**.
- **Project Information**: To find this information, select **Project** → **Status**.
- **Driver Version and Communication Log**: Displays in the Studio *Output* window when the driver is running.
- **Device Model and Boards**: Consult the hardware manufacturer's documentation for this information.

Sample Application


You will find a sample application in the `/COMMUNICATION EXAMPLES/MODBU` directory. We strongly recommend that you use this sample application to test the MODBU 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.
3. Execute the *Viewer* module in Studio to display information about the driver communication.

 **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	2.02	Roberto V. Junior	Jul/30/1999	<ul style="list-style-type: none"> First driver version Driver available for Windows CE
B	2.03	Roberto V. Junior	Dec/13/1999	Added Read and Write of Floating-point operand
C	2.04	Roberto V. Junior	Jun/5/2000	Added CRC verification of device response
D	2.05	Lourenço Teodoro	Oct/30/2000	Added MAIN DRIVER SHEET feature
E	2.06	Lourenço Teodoro	Nov/14/2001	<ul style="list-style-type: none"> Added FPS, FP3, and FP3S data type Changed Ero functionality
F	2.06	Fabiola Fantinato	Dec/3/2001	Revised document to conform to documentation standards
G	2.07	Roberto V. Junior	Feb/21/2002	<ul style="list-style-type: none"> Added bit read/write to 4x and 3x data type Fixed bug that caused 100% CPU usage when a message was waiting in a timeout situation Added a warning message in the LogWin when the AddressReference in Header field plus the lowest AddressOffset in Address column is equal to zero
H	2.08	Eric Vigiani	Aug/22/2003	<ul style="list-style-type: none"> Added DW data type
I	2.09	Eric Vigiani	Dec/11/2003	<ul style="list-style-type: none"> Implemented Signed/Unsigned option by address Added DW3 data type
J	2.10	Eric Vigiani	May/25/2004	<ul style="list-style-type: none"> Implemented writing group commands when writing FP values
K	2.11	Fábio H.Y. Komura	Jun/14/2004	<ul style="list-style-type: none"> Added DWS and DW3S data types Implemented SwapWord for FP, FPS, FP3, FP3S, DW, DWS, DW3 and DW3S Changed FP, FPS, FP3, FP3S, DW, DWS, DW3 and DW3S with and without SwapWord to conformance to standards (FPS, FP3S, DWS and DW3S are data types with Byte Swap)
L	2.12	Fábio H.Y. Komura	Sep/03/2004	Added BCD, BCD3, BCDDW, BCDDWS, BCDDW3 and BCDDW3S data type.
M	2.13	Fábio H.Y. Komura	Sep/24/2004	Fixed problem with Parser Address
N	2.14	Leandro Coeli	Jan/19/2005	Insert String type
O	2.15	Leandro Coeli	Feb/19/2005	Implemented LRC to ASCII communication
P	2.16	Leandro Coeli	Apr/26/2005	Implemented configurable Block Size