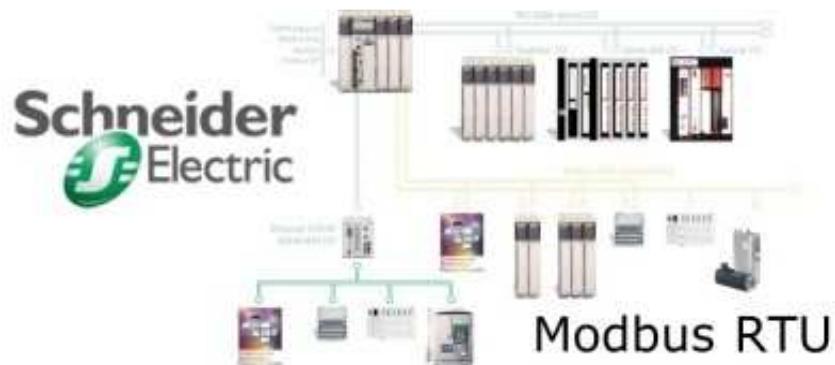


# XMODBUS Driver Manual

Gould Modicon Modbus RTU/JBus Protocol Driver



## CPKSoft Engineering

Process Monitoring and Industrial  
Automation Software

Copyright 1990-2008, CPKSoft Engineering. All rights reserved.

# Index

|           |  |           |
|-----------|--|-----------|
| <b>1.</b> | <b>Introduction</b>  | <b>3</b>  |
| <b>2.</b> | <b>Driver details</b>  | <b>4</b>  |
| 2.1.      | Driver overview .....  | 4         |
| 2.2.      | Supported devices.....   | 4         |
| <b>3.</b> | <b>Command list</b>  | <b>7</b>  |
| 3.1.      | Read Coil Status .....   | 7         |
| 3.2.      | Read Input Status.....   | 7         |
| 3.3.      | Read Holding Registers as Unsigned 16-bit Integers .....                     | 8         |
| 3.4.      | Read Holding Registers as Signed 16-bit Integers .....                       | 8         |
| 3.5.      | Read Holding Registers as Unsigned 32-bit Integers .....                     | 9         |
| 3.6.      | Read Holding Registers as Signed 32-bit Integers .....                       | 10        |
| 3.7.      | Read Holding Registers as Unsigned 32-bit Integers (words reversed) .....    | 10        |
| 3.8.      | Read Holding Registers as Signed 32-bit Integers (words reversed).....       | 11        |
| 3.9.      | Read Holding Registers as IEEE Floats .....                                  | 11        |
| 3.10.     | Read Holding Registers as IEEE Floats (bytes reversed).....                  | 12        |
| 3.11.     | Read Holding Registers as IEEE Floats (words reversed).....                  | 13        |
| 3.12.     | Read Input Registers as Unsigned 16-bit Integers.....                        | 13        |
| 3.13.     | Read Input Registers as Signed 16-bit Integers.....                          | 14        |
| 3.14.     | Read Input Registers as Unsigned 32-bit Integers.....                        | 14        |
| 3.15.     | Read Input Registers as Signed 32-bit Integers.....                          | 15        |
| 3.16.     | Read Input Registers as Unsigned 32-bit Integers (words reversed).....       | 16        |
| 3.17.     | Read Input Registers as Signed 32-bit Integers (words reversed).....         | 16        |
| 3.18.     | Read Input Registers as IEEE Floats.....                                     | 17        |
| 3.19.     | Read Input Registers as IEEE Floats (bytes reversed).....                    | 17        |
| 3.20.     | Read Input Registers as IEEE Floats (words reversed) .....                   | 18        |
| 3.21.     | Force Single Coil.....   | 19        |
| 3.22.     | Force Multiple Coils.....  | 19        |
| 3.23.     | Preset Single Register as Unsigned 16-bit Integer.....                       | 20        |
| 3.24.     | Preset Multiple Registers as Unsigned 16-bit Integers .....                  | 20        |
| 3.25.     | Preset Multiple Registers as Unsigned 32-bit Integers .....                  | 21        |
| 3.26.     | Preset Multiple Registers as Signed 32-bit Integers .....                    | 22        |
| 3.27.     | Preset Multiple Registers as Unsigned 32-bit Integers (words reversed) ..... | 22        |
| 3.28.     | Preset Multiple Registers as Signed 32-bit Integers (words reversed) .....   | 23        |
| 3.29.     | Preset Multiple Registers as IEEE Floats .....                               | 23        |
| 3.30.     | Preset Multiple Registers as IEEE Floats (bytes reversed).....               | 24        |
| 3.31.     | Preset Multiple Registers as IEEE Floats (words reversed) .....              | 25        |
| <b>4.</b> | <b>Appendices</b>  | <b>26</b> |
| 4.1.      | Error messages .....   | 26        |
| 4.2.      | Keywords list.....   | 26        |

# 1. Introduction

CPKSoft Engineering assumes no responsibility for any errors that may appear in this document. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

This driver is included with all unlimited licenses of TAS-HMITalk. It is not sold separately. It requires the TAS-HMITalk ActiveX to work, therefore it cannot be used as a stand-alone driver.

If you use this driver in your applications, you need to include the xmodbus.tlk in the set of files that you distribute. This file must be located in the same folder where the hmitalk.ocx file is registered in order to be found by the activex when the applications are executed.

The source-code for the xmodbus.tlk driver is available in plain-C language for additional USD 199 if you own a license of TAS-HMITalk 8.04 or higher.

Refer to the following link to visit the xmodbus driver page at CPKSoft Engineering website: <http://www.cpksoft.com/tabid/55/ProductID/69/PageIndex/1/Default.aspx>.

Visit this link if you want to see a complete list of drivers that are currently available for TAS-HMITak: <http://www.cpksoft.com/Drivers/tabid/55/Default.aspx>.

Also, refer to this link if you are interested in purchasing a license of the most recent version of TAS-HMITalk: <http://www.cpksoft.com/Products/tabid/54/Default.aspx>.

We welcome your comments about this document. You can reach us by e-mail at [contact @ cpksoft.com](mailto:contact@cpksoft.com).

## 2. Driver details

### 2.1. Driver overview

XMODBUS driver allows you to connect to any equipment which uses the Gould Modicon MODBUS Binary Protocol, also known as 'MODBUS RTU' (Remote Terminal Unit framing). Also supports the JBUS protocol, which is in fact a subset of the MODBUS RTU protocol.

This driver implements Modbus commands 1, 2, 3, 4, 5, 6, 15 and 16 as they are described in the Gould Modicon Modbus Protocol Manual. It also implements some special commands that are not part of the original Modbus specification that bring support for special data formats used by certain devices.

This driver expects that you connect to your device through its serial port. The device should behave as a slave in your RS-232/485 network. This driver supports serial-over-ethernet so you can alternatively use some kind of transparent ethernet/serial converter to reach your device using your LAN.

### 2.2. Supported devices

This driver can communicate with these devices, but is not necessarily limited to this list:

- ABB ION 8400 Power Meter
- ABB ION 8500 Power Meter
- ABB MASTER PIECE 90 PLC
- ABB PROCONTIC 07-KR31 PLC
- ABB PROCONTIC CS-31/07 KR-31
- ABB PROCONTIC CS-31/07 KT-31
- ABB PROCONTIC CS-31/07 KT-91 using 07 MK-92
- ABB PROCONTIC CS-31/07 KT-92 using 07 MK-92
- ABB PROCONTIC CS-31/07 KT-93 using 07 MK-92
- ACROMAG 902MB, 10 channel relay output module
- ADVANTECH ADAM-4051 16 channel digital input module
- ADVANTECH ADAM-4068 8 channel relay output module
- AEG MODICON 984-485 PLC
- AEG MODICON 984-685 PLC
- AEG MODICON 984-785 PLC
- AEG MODICON 984-A PLC
- AEG MODICON 984-B PLC
- AEG MODICON 984-X PLC
- AEG MODICON A984-120 PLC
- AEG MODICON A984-130 PLC
- AEG MODICON A984-131 PLC
- AEG MODICON A984-140 PLC

AEG MODICON A984-145 PLC  
AEG MODICON Micro 110 CPU 311 00  
AEG MODICON QUANTUM 140-113  
AEG MODICON QUANTUM 140-213  
AEG MODICON QUANTUM 140-424  
ALGODUE UPM 3060  
ARCOM CONTROL SYSTEMS M1 Mini RTU  
AREVA ALSTOM MICOM M300 Relay  
BAKER 6532 RTU  
BAKER 6532-EFM RTU  
CIRCUTOR CVM-144 Network Analyzer  
CONTROL MICROSYSTEMS TeleSafe 16EX RTU  
CONTROL MICROSYSTEMS TeleSafe Micro16 RTU  
DANIEL S500 Flow Computer  
DYNAMIC Fluid Meter  
ELLIOT Flow Computer  
E-MON ProMon (modbus version)  
E-MON Pulse Output Meters  
ENDRESS HAUSER Proline Promag 53 Electromagnetic Flowmeter  
ENDRESS HAUSER Proline Promass 83 Coriolis Flowmeter  
FREDRIKSSON Autolog AL-16 PLC  
FREDRIKSSON Autolog AL-2000 PLC  
FREDRIKSSON Autolog AL-32 PLC  
GENERAL ELECTRIC GE EPM 2000 Power Meter  
GENERAL ELECTRIC GE kV2c with Modbus card  
IEA S6800 4-Digit Displays  
IEA S6800 6-Digit/1-Setpoint Displays  
IEA S6800 6-Digit/3-Setpoint Displays  
IEA S9800 Single-Loop Controller  
ITT BARTON 1140 Flow Meter  
JANITZA ELECTRONICS UMG 503 Measuring Device  
KOYO Direct Logic DL205 PLC  
KOYO Direct Logic DL305 Series  
KOYO Direct Logic DL330 PLC  
KOYO Direct Logic DL330P PLC  
KOYO Direct Logic DL340 PLC  
KOYO Direct Logic DL405 Series  
KOYO Direct Logic DL430 PLC  
KOYO Direct Logic DL440 PLC  
MAGNETEK GPD 515 Drive  
MULTILIN SR735/SR737 Feeder Relay  
OBVIUS A8923-4 IO Module (4A4P)  
OBVIUS R9120 ModHopper  
OMNI Flow Computer  
PDI BCMS  
POWER MEASUREMENT 3300 ACM Power Meter  
POWER MEASUREMENT 3710 ACM Power Meter  
POWER MEASUREMENT 7300 ACM Power Meter  
POWER MEASUREMENT ION 6200

POWER MEASUREMENT ION 6300  
POWER MEASUREMENT ION 7300, 7330, 7350  
POWER MEASUREMENT ION 7500, 7600  
POWER MEASUREMENT ION 8400 Power Meter  
POWER MEASUREMENT ION 8500  
POWER MEASUREMENT ION 8500 Power Meter  
SCHENK Vibration Meters  
SCHNEIDER ELECTRIC SR760 PROTECTION RELAY  
SEEKIRK Annunciator A1700  
SIEMENS ION 9200  
SIEMENS ION 9300, 9330, 9350  
SIEMENS ION 9500, 9600  
SIEMENS MICRO PLC S7-214 with Special Software  
SPIRAX-SARCO NOVUS N-2000 PID Controller  
SQUARE-D POWER LOGIC Enercept power meter class 3020  
SQUARE-D POWER LOGIC Energy Meter with comm board.  
SQUARE-D POWER LOGIC PM710  
TECPET MNT02 Tank Level Meter  
TECPET MNT06 Tank Level Meter  
TELEMECANIQUE TSX 17-20 SCG 1131/1161 PLC  
TELEMECANIQUE TSX 3722 PLC  
TEXAS INSTRUMENTS TI-330 PLC Using 01DM Module  
TEXAS INSTRUMENTS TI-335 PLC Using 01DM Module  
TEXAS INSTRUMENTS TI-425 PLC Using 01DM Module  
TEXAS INSTRUMENTS TI-430 PLC Using 01DM Module  
TEXAS INSTRUMENTS TI-435 PLC Using 01DM Module  
VERIS Enercept H8035 and H8036  
VERIS H663 and H704 Branch Current Monitor  
VERIS H8163 Energy Meter with comm board  
VERIS H8238 Multi Circuit Monitor  
VERIS H8436, H8437 Power Meter  
VERIS HXO/T-485M

## 3. Command list

### 3.1. Read Coil Status

---

**Description of this command:**

Obtains current status (ON/OFF) in a group of logic coils. This command implements Modbus function 1.

**Type of data handled by this command:**

Digital Input

**Number of points accepted by this command:**

1-1000

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

1

**Meaning of the DriverP2 parameter:**

Indicates the starting coil address.

**Values that are returned:**

Value in PointValue (0) = First coil status (0=OFF, 1=ON)  
Value in PointValue (1) = Second coil status (0=OFF, 1=ON)  
...  
Value in PointValue (n-1) = Last coil status (0=OFF, 1=ON)

### 3.2. Read Input Status

---

**Description of this command:**

Obtains current status (ON/OFF) in a group of discrete inputs. This command implements Modbus function 2.

**Type of data handled by this command:**

Digital Input

**Number of points accepted by this command:**

1-1000

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

2

**Meaning of the DriverP2 parameter:**

Indicates the starting input address.

**Values that are returned:**

Value in PointValue (0) = First input status (0=OFF, 1=ON)

Value in PointValue (1) = Second input status (0=OFF, 1=ON)

...

Value in PointValue (n-1) = Last input status (0=OFF, 1=ON)

### 3.3. Read Holding Registers as Unsigned 16-bit Integers

---

**Description of this command:**

Obtains the current values in one or more holding registers as unsigned 16-bit integers from 0 to 65535. This command implements Modbus function 3.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-125

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

3

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value (0-65535)

Value in PointValue (1) = Second register value (0-65535)

...

Value in PointValue (n-1) = Last register value (0-65535)

### 3.4. Read Holding Registers as Signed 16-bit Integers

---

**Description of this command:**

Obtains the current values in one or more holding registers as signed 16-bit integers from -32768 to 32767. This command implements Modbus function 3.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-125

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

74

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value (-32768 to 32767)

Value in PointValue (1) = Second register value (-32768 to 32767)

...

Value in PointValue (n-1) = Last register value (-32768 to 32767)

## 3.5. Read Holding Registers as Unsigned 32-bit Integers

---

**Description of this command:**

Obtains the current values in one or more holding registers as unsigned 32-bit integer numbers. This command uses Modbus function 3.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

76

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value

Value in PointValue (1) = Second register value

...

Value in PointValue (n-1) = Last register value

## 3.6. Read Holding Registers as Signed 32-bit Integers

---

**Description of this command:**

Obtains the current values in one or more holding registers as signed 32-bit integer numbers. This command uses Modbus function 3.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

75

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

## 3.7. Read Holding Registers as Unsigned 32-bit Integers (words reversed)

---

**Description of this command:**

Obtains current values in one or more holding registers as unsigned 32-bit integer numbers with words in reverse order. This command uses Modbus function 3.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

86

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

### 3.8. Read Holding Registers as Signed 32-bit Integers (words reversed)

---

**Description of this command:**

Obtains current values in one or more holding registers as signed 32-bit integer numbers with words in reverse order. This command uses Modbus function 3.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

85

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

### 3.9. Read Holding Registers as IEEE Floats

---

**Description of this command:**

Obtains the current values in one or more holding registers as 32-bit IEEE floating-point values. This command uses Modbus function 3.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

66

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

## 3.10. Read Holding Registers as IEEE Floats (bytes reversed)

---

**Description of this command:**

Obtains the current values in one or more holding registers as 32-bit IEEE floating-point values assuming that bytes are received in reverse order. This command uses Modbus function 3.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

68

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

## 3.11. Read Holding Registers as IEEE Floats (words reversed)

---

**Description of this command:**

Obtains current values in one or more holding registers as 32-bit IEEE floating point values assuming that the words order is received in reversed order. This command uses Modbus function 3.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

80

**Meaning of the DriverP2 parameter:**

Indicates the starting memory address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

**Important note:**

This command was added to support the floating-point format used by the TSX-3722 PLC from Telemecanique.

## 3.12. Read Input Registers as Unsigned 16-bit Integers

---

**Description of this command:**

Obtains the current values in one or more input registers as unsigned 16-bit integers from 0 to 65535. This command implements Modbus function 4.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-125

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

4

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value (0-65535)

Value in PointValue (1) = Second register value (0-65535)

...

Value in PointValue (n-1) = Last register value (0-65535)

### 3.13. Read Input Registers as Signed 16-bit Integers

---

**Description of this command:**

Obtains the current values in one or more input registers as signed 16-bit integers from -32768 to 32767. This command implements Modbus function 4.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-125

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

179

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value (-32768 to 32767)

Value in PointValue (1) = Second register value (-32768 to 32767)

...

Value in PointValue (n-1) = Last register value (-32768 to 32767)

### 3.14. Read Input Registers as Unsigned 32-bit Integers

---

**Description of this command:**

Obtains the current values in one or more input registers as unsigned 32-bit integer numbers. This command uses Modbus function 4.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

176

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

---

## 3.15. Read Input Registers as Signed 32-bit Integers

---

**Description of this command:**

Obtains the current values in one or more input registers as signed 32-bit integer numbers.  
This command uses Modbus function 4.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

175

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

## 3.16. Read Input Registers as Unsigned 32-bit Integers (words reversed)

---

**Description of this command:**

Obtains current values in one or more input registers as unsigned 32-bit integer numbers with words in reverse order. This command uses Modbus function 4.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

186

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

## 3.17. Read Input Registers as Signed 32-bit Integers (words reversed)

---

**Description of this command:**

Obtains current values in one or more input registers as signed 32-bit integer numbers with words in reverse order. This command uses Modbus function 4.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

185

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

## 3.18. Read Input Registers as IEEE Floats

---

**Description of this command:**

Obtains the current values in one or more input registers as 32-bit IEEE floating-point values. This command uses Modbus function 4.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

166

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

## 3.19. Read Input Registers as IEEE Floats (bytes reversed)

---

**Description of this command:**

Obtains the current values in one or more input registers as 32-bit IEEE floating-point values assuming that bytes are received in reverse order. This command uses Modbus function 4.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

168

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

## 3.20. Read Input Registers as IEEE Floats (words reversed)

---

**Description of this command:**

Obtains current values in one or more input registers as 32-bit IEEE floating point values assuming that the words order is received in reversed order. This command uses Modbus function 4.

**Type of data handled by this command:**

Analog Input

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station number (0-255).

**Meaning of the DriverP1 parameter:**

180

**Meaning of the DriverP2 parameter:**

Indicates the starting memory address.

**Values that are returned:**

Value in PointValue (0) = First register value  
Value in PointValue (1) = Second register value  
...  
Value in PointValue (n-1) = Last register value

## 3.21. Force Single Coil

---

**Description of this command:**

Forces a single logic coil to a state ON or OFF. This command implements Modbus function 5.

**Type of data handled by this command:**

Digital Output

**Number of points accepted by this command:**

1

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

5

**Meaning of the DriverP2 parameter:**

Indicates the coil address.

**Values that are sent:**

Value in PointValue (0) = New coil status (0=OFF, 1=ON)

## 3.22. Force Multiple Coils

---

**Description of this command:**

Forces a series of consecutive logic coils to selected ON or OFF states. This command implements Modbus function 15.

**Type of data handled by this command:**

Digital Output

**Number of points accepted by this command:**

1-1000 (It is convenient for the groups to be a multiple of 8 elements)

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

15

**Meaning of the DriverP2 parameter:**

Indicates the starting coil address.

**Values that are sent:**

Value in PointValue (0) = New status for first coil (0=OFF, 1=ON)

Value in PointValue (1) = New status for second coil (0=OFF, 1=ON)

...

Value in PointValue (n-1) = New status for last coil (0=OFF, 1=ON)

## 3.23. Preset Single Register as Unsigned 16-bit Integer

---

**Description of this command:**

Places a specified value into a holding register as an unsigned 16-bit integer value from 0 to 65535. This command implements Modbus function 6.

**Type of data handled by this command:**

Analog Output

**Number of points accepted by this command:**

1

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

6

**Meaning of the DriverP2 parameter:**

Indicates the register address.

**Values that are sent:**

Value in PointValue (0) = New register value (0-65535)

## 3.24. Preset Multiple Registers as Unsigned 16-bit Integers

---

**Description of this command:**

Places specified values into a series of consecutive holding registers, as unsigned 16-bit integer values from 0 to 65535. This command implements Modbus function 16.

**Type of data handled by this command:**

Analog Output

**Number of points accepted by this command:**

1-125

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

16

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are sent:**

Value in PointValue (0) = New value for first register (0-65535)

Value in PointValue (1) = New value for second register (0-65535)

...

Value in PointValue (n-1) = New value for last register (0-65535)

## 3.25. Preset Multiple Registers as Unsigned 32-bit Integers

---

**Description of this command:**

Places specified values into a series of consecutive holding registers, as unsigned 32-bit integer values. This command uses Modbus function 16.

**Type of data handled by this command:**

Analog Output

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

78

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are sent:**

Value in PointValue (0) = New value for first register

Value in PointValue (1) = New value for second register

...

Value in PointValue (n-1) = New value for last register

## 3.26. Preset Multiple Registers as Signed 32-bit Integers

---

**Description of this command:**

Places specified values into a series of consecutive holding registers, as signed 32-bit integer values. This command uses Modbus function 16.

**Type of data handled by this command:**

Analog Output

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

77

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are sent:**

Value in PointValue (0) = New value for first register  
Value in PointValue (1) = New value for second register  
...  
Value in PointValue (n-1) = New value for last register

## 3.27. Preset Multiple Registers as Unsigned 32-bit Integers (words reversed)

---

**Description of this command:**

Places specified values into a series of consecutive holding registers, as unsigned 32-bit integer values with words in reverse order. This command uses Modbus function 16.

**Type of data handled by this command:**

Analog Output

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

88

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are sent:**

Value in PointValue (0) = New value for first register  
Value in PointValue (1) = New value for second register  
...  
Value in PointValue (n-1) = New value for last register

## 3.28. Preset Multiple Registers as Signed 32-bit Integers (words reversed)

---

**Description of this command:**

Places specified values into a series of consecutive holding registers, as signed 32-bit integer values with words in reverse order. This command uses Modbus function 16.

**Type of data handled by this command:**

Analog Output

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

87

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are sent:**

Value in PointValue (0) = New value for first register  
Value in PointValue (1) = New value for second register  
...  
Value in PointValue (n-1) = New value for last register

## 3.29. Preset Multiple Registers as IEEE Floats

---

**Description of this command:**

Places specified values into a series of consecutive holding registers as 32-bit IEEE floating point values. This command uses Modbus function 16.

**Type of data handled by this command:**

Analog Output

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

70

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are sent:**

Value in PointValue (0) = New value for first register  
Value in PointValue (1) = New value for second register  
...  
Value in PointValue (n-1) = New value for last register

### **3.30. Preset Multiple Registers as IEEE Floats (bytes reversed)**

---

**Description of this command:**

Places specified values into a series of consecutive holding registers as 32-bit IEEE floating point values where the bytes order is reversed. This command uses Modbus function 16.

**Type of data handled by this command:**

Analog Output

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

72

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are sent:**

Value in PointValue (0) = New value for first register  
Value in PointValue (1) = New value for second register

...  
Value in PointValue (n-1) = New value for last register

### 3.31. Preset Multiple Registers as IEEE Floats (words reversed)

---

**Description of this command:**

Places specified values into a series of consecutive holding registers as 32-bit IEEE floating point values where the required words order must be reversed. This command uses Modbus function 16.

**Type of data handled by this command:**

Analog Output

**Number of points accepted by this command:**

1-62

**Meaning of the DriverP0 parameter:**

Station Number (0-255). If the station number is 0, the command is sent as a broadcast message and no response is expected.

**Meaning of the DriverP1 parameter:**

81

**Meaning of the DriverP2 parameter:**

Indicates the starting register address.

**Values that are sent:**

Value in PointValue (0) = New value for first register  
Value in PointValue (1) = New value for second register  
...  
Value in PointValue (n-1) = New value for last register

**Important note:**

This command was added to support the format used by the TSX-3722 PLC from Telemecanique.

## 4. Appendices

### 4.1. Error messages

The following list shows all the possible error messages that can be returned by the protocol driver during a failed communication in the 'DriverStatus' property.

This list does not include some error messages that can be returned by the activex component while attempting to establish a connection.

- [1005] DRIVER (Internal): Invalid driver stage
- [1300] PROTOCOL (Timeout): No answer
- [1421] PROTOCOL (Format): Negative acknowledge received from device
- [2147] CONFIG (NumValues): Only one value can be read or written
- [2185] CONFIG (NumValues): Too many values (max=125)
- [2229] CONFIG (NumValues): Too many values (max=62)
- [3014] CONFIG (P0): Invalid device address (0-255)
- [3022] CONFIG (P0): Invalid device address (1-255)
- [3508] CONFIG (P1): Invalid command
- [4001] CONFIG (P2): Invalid address
- [8013] CONFIG (Remote): Acknowledge
- [8034] CONFIG (Remote): Busy (rejected message)
- [8138] CONFIG (Remote): Failure in associated device
- [8168] CONFIG (Remote): Illegal data address
- [8170] CONFIG (Remote): Illegal data value
- [8172] CONFIG (Remote): Illegal function
- [8347] CONFIG (Remote): Unknown error

### 4.2. Keywords list

The following list shows a set of words directly related to this driver.

"01DM, 07KR31, 110, 1131, 1140, 1161, 140-113, 140-213, 140-424, 16EX, 17-20, 1Setpoint, 2000, 3020, 3060, 311, 3300, 3710, 3722, 3Setpoint, 4A4P, 4Digit, 503, 515, 6200, 6300, 6532, 6532EFM, 6Digit, 7300, 7330, 7350, 7500, 7600, 8400, 8500, 902MB, 9200, 9300, 9330, 9350, 9500, 9600, 984-485, 984-685, 984-785, 984A, 984B, 984X, A1700, A8923-4, A984-120, A984-130, A984-131, A984-140, A984-145, ABB, ACM, ACROMAG, ADAM4051, ADAM4068, ADVANTECH, AEG, AL16, AL2000, AL32, ALGODUE, ALSTOM, Analyzer, Annunciator, ARCOM, AREVA, Autolog, BAKER, BARTON, BCMS, board., Branch, CIRCUTOR, CONTROL, Controller, Coriolis, CPU, CS31/07, Current, CVM144, DANIEL, Direct, Displays, DL205, DL305, DL330, DL330P, DL340, DL405, DL430, DL440, Drive, DYNAMIC, E-MON, ELECTRIC, Electromagnetic, ELECTRONICS, ELLIOT, ENDRESS, Enercept, Energy, EPM, Feeder, Flow, Flowmeter, Fluid, FREDRIKSSON, GE, GENERAL, Gould, GPD, H663, H704, H8035, H8036, H8163, H8238, H8436, H8437, HAUSER, HXO, IEA, INSTRUMENTS,

IO, ION, ITT, JANITZA, JBus, KOYO, KR31, KT31, KT91, KT92, KT93, kV2c, Level, Logic, M300, MAGNETEK, MASTER, MEASUREMENT, Measuring, Meter, Meters, MICOM, Micro, Micro16, MICROSYSTEMS, Mini, MK92, MNT02, MNT06, modbus, ModHopper, MODICON, Monitor, Multi, MULTILIN, N2000, Network, NOVUS, OBVIUS, OMNI, PDI, PID, PIECE, PLC, PM710, Power, PROCONTIC, Proline, Promag, Promass, ProMon, PROTECTION, Pulse, QUANTUM, R9120, relay, RTU, S500, S6800, S7-214, S9800, SCG, SCHENK, SCHNEIDER, SEEKIRK, Series, SIEMENS, Single-Loop, SPIRAX-SARCO, SQUARE-D, SR735, SR737, SR760, SYSTEMS, T485M, Tank, TECPET, TELEMECANIQUE, TeleSafe, TEXAS, TI330, TI335, TI425, TI430, TI435, TSX, UMG, UPM, VERIS, version, Vibration".