

ADM200

Modbus Communications

18th September 2020 | Version 01



COPYRIGHT AND CONFIDENTIAL

© ALL RIGHTS RESERVED. This Document and the intellectual property contained herein are the property of Trumeter. No part of this document may be disclosed, reproduced, distributed or transmitted in any form or by any means without prior written permission.

NOTICE

Printed documents and copies stored outside of Trumeter and are uncontrolled. For the latest version please refer to the company's document register. This document may refer to documents whose ownership is outside that of Trumeter. The ownership and right to copyright of these documents are acknowledged. No part of these documents may be used without the prior written permission of the document owners.

Table of Contents

1	About this document	3
2	ADM Configuration: Modbus RTU.	3
2.1	Slave ID	3
2.2	Baud Rate	3
2.3	Stop Bits & Parity	4
2.4	Modbus RTU function Code Support	4
3	ADM Configuration: Modbus TCP.	5
3.1	ModBus Enable	5
3.2	DHCP Client	5
3.3	Configurable options	5
3.4	Modbus TCP function Code Support	6
4	Register Map for ModBus RTU & ModBus TCP	6
4.1	Modbus sentence protocol	6
5	Modbus Communication Examples	7
5.1	Modbus RTU communication example	7
5.2	Modbus TCP communication example	8
		8

1 About this document

This document describes the use of ModBus in the ADM-MAX.

ModBus is implemented in the ADM-MAX to enable the reading of the measured value and the status of the alarms. Configuration of the ADM-MAX measurements can be performed by using the ADM-MAX configurator. This document defines the configuration and usage of the ModBus features of the ADM-MAX for both ModBus RTU and ModBus TCP. The readable registers are common to both ModBus communication methods.

By default both the ModBus RTU and ModBus TCP/IP interfaced are enabled. To configure, please use the ADM-MAX configurator which is available from <https://www.trumeter.com/ADM-software-download/>

2 ADM Configuration: Modbus RTU.

2.1 Slave ID

Any value between 0 and 255 can be entered.

2.2 Baud Rate

The following baud rates are supported.

Baud Rate					
4800	9600	14400	19200	24000	28800
33600	38400	43200	48000	52800	57600
62400	67200	72000	76800	81600	86400
91200	96000	100800	105600	110400	115200
120000	124800	129600	134400	139200	144000
148800	153600	158400	163200	168000	172800
177600	182400	187200	192000		

2.3 Stop Bits & Parity

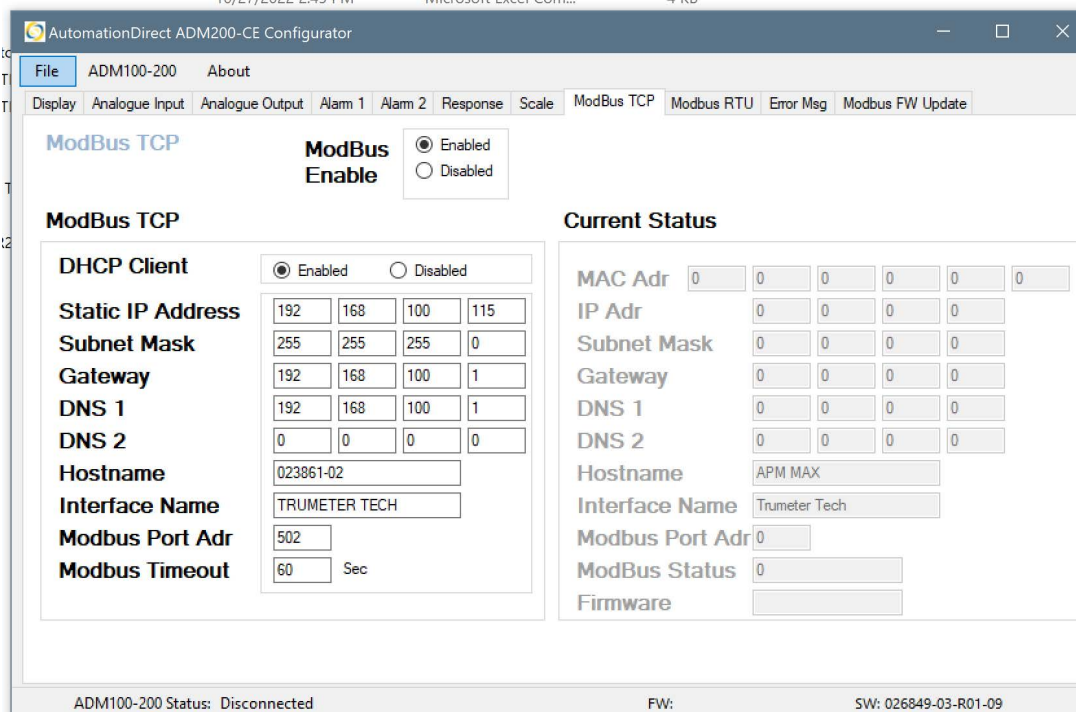
The following combinations of Stop bits, and Parity bits are supported.

- 2 Stop bits and No Parity bits
- 1 Stop bit and an Even Parity bit.
- 1 Stop Bit and an Odd Parity bit.

2.4 Modbus RTU function Code Support

Function Code 4 – “Read Input Register” is supported.

3 ADM Configuration: Modbus TCP.



3.1 ModBus Enable

Enable and disable are selectable.

3.2 DHCP Client

Enable and disable are selectable.

3.3 Configurable options

The following parameters are adjustable. IP address and Subnet Mask are only adjustable if DHCP is disabled.

Parameter
IP address
Subnet mask
Gateway
DNS 1
DNS 2
Hostname
Interface Name
Modbus port address
Modbus timeout period

3.4 Modbus TCP function Code Support

Function Code 4 – “Read Input Register” is supported.

4 Register Map for ModBus RTU & ModBus TCP

There are 6 readable registers available from the ADM-MAX:

Input registers		
Address	Type	Details
1 - 2	32bit Float	Displayed Value
3 - 4	32bit Float	Measured Value, before user offsets are applied
5	16bit int	Alarm 1 Status
6	16bit int	Alarm 2 Status

4.1 Modbus sentence protocol

High byte (Big Endian) and High word (Big Endian). The most significant word is sent first. Each 16 bit word is sent as two 8 bit bytes, the most significant byte is sent first.

5 Modbus Communication Examples

In the following example SimplyModbus (<https://www.simplymodbus.ca>) has been used to communicate with the ADM-MAX.

Send: 0A 04 01 00 06 (request values in 6 - 16 BIT registers)
Displayed Value: 41 00 14 8E (Hex) 8.00501823
Measured Value with no offset: 41 00 14 8E (Hex) 8.00501823
Alarm 1: Active
Alarm 2: Not Active

5.1 Modbus RTU communication example

The screenshot shows the 'Simply Modbus Master 8.1.1' interface. The 'mode' is set to 'RTU', 'COM port' is '5', 'baud' is '9600', 'data bits' is '8', 'stop bits' is '2', and 'parity' is 'None'. The 'Slave ID' is '10', 'First Register' is '1', and 'No. of Regs' is '6'. The 'function code' is '4' (Read Holding Registers), and 'minus offset' is '0'. The 'register size' is set to '16 bit registers'. The 'Request' field shows the hex string '0A 04 00 01 00 06 20 B3'. The 'Response' field shows '0A 04 0C 41 00 14 8E 41 00 14 8E 00 01 00 00 B0 9D'. A table on the right displays the results of the read operation:

copy down	register #	bytes	results	notes	clear notes
32bit Float	1	4100 148E	8.00501823	Displayed Value	
32bit Float	3	4100 148E	8.00501823	MV no offset	
16bit INT	5	0001	1	Alarm 1	
16bit INT	6	0000	0	Alarm 2	

At the bottom, a log window shows the following exchange:

```

2020/09/03 09:53:36 >>> 0A 04 00 01 00 06 20 B3
2020/09/03 09:53:36 < 0A 04 0C 41 00 14 8E 41 00 14 8E 00 01 00 00 B0 9D
    
```

5.2 Modbus TCP communication example

Simply Modbus TCP Client 8.1.0

mode: TCP IP Address: 10.0.4.46 Port: 502

Slave ID: 10 First Register: 1 No. of Regs: 6

function code: 4 minus offset: 0 register size: 16 bit registers

Request: 00 02 00 00 00 06 0A 04 00 01 00 06

Response: 00 01 00 00 00 0F 0A 04 0C 41 00 15 0F 41 00 15 0F 00 01 00 00

copy down	register#	bytes	results	notes
32bit Float	1	4100 150F	8.00514126	Displayed Value
32bit Float	3	4100 150F	8.00514126	MV no offset
16bit INT	5	0001	1	Alarm1
16bit INT	6	0000	0	Alarm 2

Request: 00 02 00 00 00 06 0A 04 00 01 00 06

Response: 00 01 00 00 00 0F 0A 04 0C 41 00 15 0F 41 00 15 0F 00 01 00 00

Request: 00 02 00 00 00 06 0A 04 00 01 00 06

Response: 00 01 00 00 00 0F 0A 04 0C 41 00 15 0F 41 00 15 0F 00 01 00 00

2020/09/03 09:53:07 >>> 00 01 00 00 00 06 0A 04 00 01 00 06

2020/09/03 09:53:07 <<< 00 01 00 00 00 0F 0A 04 0C 41 00 15 0F 41 00 15 0F 00 01 00 00