

• OXYGEN Modbus TCP Plugin

TECHNICAL REFERENCE MANUAL

WELCOME TO THE WORLD OF DEWETRON!

Congratulations on your new device! It will supply you with accurate, complete and reproducible measurement results for your decision making.

Look forward to the easy handling and the flexible and modular use of your DEWETRON product and draw upon more than 30 years of DEWETRON expertise in measurement engineering.



 $\mathbf{\nabla}$



THE MEASURABLE DIFFERENCE.

The information contained in this document is subject to change without notice.

DEWETRON GmbH (DEWETRON) shall not be liable for any errors contained in this document. DEWETRON MAKES NO WARRANTIES OF ANY KIND ABOUT THIS DOCUMENT, WHETHER EXPRESS OR IMPLIED. DEWETRON SPECIFICALLY DISCLAIMS THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

DEWETRON shall not be liable for any direct, indirect, special, incidental, or consequential damages, whether based on contract, tort, or any other legal theory, in connection with the furnishing of this document or the use of the information in this document.

Technical Support

Please contact your local authorized DEWETRON representative first for any support and service questions.

For Asia and Europe, please contact:

DEWETRON GmbH Parkring 4 8074 Grambach AUSTRIA

 Tel.:
 +43 316 3070

 Fax:
 +43 316 307090

 Email:
 support@dewetron.com

 Web:
 http://www.dewetron.com

The telephone hotline is available Monday to Friday between 08:00 and 17:00 CET (GMT +1:00)

Restricted Rights Legend:

Use Austrian law for duplication or disclosure.

DEWETRON GmbH Parkring 4 8074 Grambach AUSTRIA

Printing History:

Please refer to the page bottom for printing version. Copyright © DEWETRON GmbH

This document contains information which is protected by copyright. All rights are reserved. Reproduction, adaptation, or translation without prior written permission is prohibited, except as allowed under the copyright laws.

All trademarks and registered trademarks are acknowledged to be the property of their owners.

Before updating your software please contact DEWETRON. Use only original software from DEWETRON.

Please find further information at <u>www.dewetron.com</u>.

For America, please contact:

DEWETRON, Inc. 2850 South County Trail, Unit 1 East Greenwich, RI 02818 U.S.A.

Tel.:	+1 401 284 3750
Toll-free:	+1 877 431 5166
Fax:	+1 401 284 3755
Email:	us.support@dewetron.com
Web:	http://www.dewetron.com

The telephone hotline is available Monday to Friday between 08:00 and 17:00 GST (GMT -5:00)

Table of Content

1	PREFA	ACE	. 4
2	FUNC	TIONAL OVERVIEW	. 5
	2.1 2.2 2.3 2.4	Concept 5 Datatypes and scaling Constant Linear Scaling Sunspec Scaling	5.5
3	GETT	ING STARTED	. 7
4	XML	CONFIG FILE	. 8
	4.1 4.2 4.2.1	MINIMUM WORKING EXAMPLE PROPERTIES 9 TCPENDPOINT	. 8
	4.2.2 4.2.3 4.2.4 4.2.5 4.2.6	Device 9 ModbusDevice Register 10 Datatypes	9 10

1 PREFACE

This documentation describes, how to use the MODBUS Plugin in OXYGEN.

From Wikipedia [https://en.wikipedia.org/wiki/Modbus]

Modbus is a data communications protocol for use with its programmable logic controllers (PLCs). Modbus has become a de facto standard communication protocol and is now a commonly available means of connecting industrial electronic devices. Modbus is popular in industrial environments because it is openly published and royalty-free. It was developed for industrial applications, is relatively easy to deploy and maintain compared to other standards, and places few restrictions - other than the datagram (packet) size - on the format of the data to be transmitted. Modbus uses the RS485 or Ethernet as its wiring type. Modbus supports communication to and from multiple devices connected to the same cable or Ethernet network. For example, a device that measures temperature and a different device to measure humidity, both of which communicates the measurements to a computer.

2 FUNCTIONAL OVERVIEW

The MODBUS Plugin for OXYGEN is an extension for the popular OXYGEN measurement software to read data from Modbus devices. This allows the user, to use Modbus devices as data sources.

Features:

- Reading Data from Modbus TCP Devices
- Support of various data types (int, float, ...)
- Support of different scaling modes (linear and sunspec)
- Independent definition of register and endpoints for simple reuse
- Per endpoint definable refresh rate

Known Limitation:

• No support of RTU

2.1 CONCEPT

- TCP Endpoint: An Endpoint is a Node with a unique IP address and port an can hold one to many devices.
- Device: Is the representation of a physical modbus device. It is described by its UNIT_ID.

2.2 DATATYPES AND SCALING

The Plugin supports two different types of data scaling:

- const_linear: Linear scaling with factor and offset
- sunssf: Variable scaling with another register value, compatible with the sunspec scaling (INT+SF)

2.3 CONSTANT LINEAR SCALING

Typically, the linear scaling is used for converting the binary data to physical values.

Output Value = read_value * scale + offset

Example:

```
read_value = 2314
scale = 0.1
offset = 0
output value = 2314 * 0.1 + 0.0 = 231.4
```

2.4 SUNSPEC SCALING

Within Solar inverters, the sunssf scaling is often used. It is defined by Sunspec. In this case, a separate register is read, which holds the actual scaling value. This specific value is an exponent of 10.

Output Value = read_value * 10^scale_reg_val

Example:

read_value = 2314
scale_reg_val = -1
output_value = 2314 * 10^-1 = 231.4

3 GETTING STARTED

- 1. Start OXYGEN (if not already started)
- 2. Open the channel list
- 3. Press "+" add channel
- 4. Select "MODBUS" in the Receiver section

-		DEWE2-PA7	1	2	3 4	5	6	7				
	LocalNode	Add Channel	- Modb	us Rece	iver							
		Constant Per	centage	Bandwid	ith	Modb	us con	nfigur	ration file		Browse	
0		(CPB) Analys	is			Confi	guratio	on file	le is valid and contains 1 device configurations.			
		Data Sources				The f	ollowir	ng 1 u	units will be used:		Add new TCP endpoint	
		Ethernet Rec	eiver			B×	<	>				
~		Modbus Rece	eiver					1.	Host 192.168.1.100	ſ		
		Example Plu	gin: WA\	/ file repla	ау		~	M	lyEndpoint Port 502	l	Add new unit	
		Example Plu replay	gin: Sing	gle matrix	file			Ur	nit MyDevice1 Unit id 1		MyDeviceType	
A		Example Plu	gin: Sim	ple mess	age						Channels: All	
	Analog	Fxample Plu	gin: Sim	ple file re	play							»
		Example Plus	gin: Sim	ple async	file							Range
(i)	└ v LocalNo	replay										
		Dynamomet	er									
		Protocols										-100 V 100 V
	A	MIL-STD-1553	3 Decod	er								-100 V 100 V
	A AI	ARINC Decod	ler									-100 V 100 V
		Data Transfer										-100 V 100 V
		Ethernet Sen	der									-100 V 100 V
		c										
	A	4								Cancel	Add	-100 V 100 V
	A	2/2 3111		TRION-160	3-11/-6-BNC			٥	Voltage		10000 Hz	-100 V 100 V
		l 2/2 Sim				-		*	-59 999089 AVGI			
	+	-	È									

- 5. Load a XML Document with the Device and Network description
- 6. Adapt settings if necessary (e.g. load only selected channels, change IP-address)
- 7. Click "Add"

4 XML CONFIG FILE

Most of the configuration is provided via a XML document. It is designed, to be very flexible in usage as well as easy to create.

4.1 MINIMUM WORKING EXAMPLE

This example consists of one TCP endpoint, one Modbus device with one register.

```
<Modbus version="1.0">

<TCPEndpoint name = "MyEndpoint" host = "192.168.1.100" port = "502" polling_rate = "1Hz">

<Device name = "MyDevice1" device_type = "MyDeviceType" unit_id = "1"/>

</TCPEndpoint>

<ModbusDevice name="MyDeviceType">

<HoldingRegister address="40000" description="This is a Test Value" name="Value 1"

type="float"/>

</ModbusDevice>

</ModbusDevice>
```

The TCPEndpoint Node in the XML describes an endpoint with the name "MyEndpoint". The host can be an ip-address or a hostname, the ip-port is 502 by default. In this case, the polling rate is set to 1Hz, which means, that the register values are read once per second. An Endpoint consists of one or more devices, which are separated by their UNIT_ID. The Endpoint is the dedicated device itself, which is holding the Modbus registers. Typically, the Endpoint is equal to the device. But in some cases, an Endpoint consists of 2 or more devices (e.g. a gateway). The important value here is the device_type. It holds the name of the device, which is defined in the bottom part of the XML (ModbusDevice). This allows the user, to re-use the description of one device in multiple Endpoints.

```
<ModbusDevice name="MyDeviceType">
<HoldingRegister address="40000" description="This is a Test Value" name="Value 1"
type="float"/>
</ModbusDevice>
```

Attention: Continuous register numbering is necessary for fast one-block retrieval. If an address is omitted in the register numbering, OXYGEN divides it into individual blocks, and is then not interpreted as a single block. By filling in the unneeded registers, the query is interpreted by OXYGEN as one block. This can also be done by filling the unneeded registers with dummy values. In the following example, Oxygen would store 3 individual blocks.

```
<ModbusDevice name="MyDeviceType">
   <HoldingRegister address="0" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="1" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="5" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="5" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="6" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="6" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="6" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="8" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="8" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="13" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="13" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="14" description=" Value" name="Value 1" type="float"/>
   <HoldingRegister address="15" description=" Value" name="Value 1" type="float"/>
```

To prevent this and to query a single block it is necessary to fill the missing addresses.

<modbusdevice <="" name="N</th><th>MyDeviceType" th=""><th>></th><th></th><th></th></modbusdevice>	>			
<holdingregister< td=""><td>address="0"</td><td>description="</td><td>Value"</td><td><pre>name="Value 1" type="float"/></pre></td></holdingregister<>	address="0"	description="	Value"	<pre>name="Value 1" type="float"/></pre>
<holdingregister< td=""><td>address="1"</td><td>description="</td><td>Value"</td><td><pre>name="Value 1" type="float"/></pre></td></holdingregister<>	address="1"	description="	Value"	<pre>name="Value 1" type="float"/></pre>
<holdingregister< td=""><td>address="2"</td><td>description="</td><td>Value"</td><td><pre>name="Value 1" type="float"/></pre></td></holdingregister<>	address="2"	description="	Value"	<pre>name="Value 1" type="float"/></pre>
<holdingregister< td=""><td>address="3"</td><td>description="</td><td>dummy"</td><td><pre>name="dummy_3" type="float"/></pre></td></holdingregister<>	address="3"	description="	dummy"	<pre>name="dummy_3" type="float"/></pre>
<holdingregister< td=""><td>address="4"</td><td>description="</td><td>dummy"</td><td><pre>name="dummy_4" type="float"/></pre></td></holdingregister<>	address="4"	description="	dummy"	<pre>name="dummy_4" type="float"/></pre>
<holdingregister< td=""><td>address="5"</td><td>description="</td><td>Value"</td><td><pre>name="Value 1" type="float"/></pre></td></holdingregister<>	address="5"	description="	Value"	<pre>name="Value 1" type="float"/></pre>
<holdingregister< td=""><td>address="6"</td><td>description="</td><td>Value"</td><td><pre>name="Value 1" type="float"/></pre></td></holdingregister<>	address="6"	description="	Value"	<pre>name="Value 1" type="float"/></pre>
<holdingregister< td=""><td>address="7"</td><td>description="</td><td>Value"</td><td><pre>name="Value 1" type="float"/></pre></td></holdingregister<>	address="7"	description="	Value"	<pre>name="Value 1" type="float"/></pre>
<holdingregister< td=""><td>address="8"</td><td>description="</td><td>Value"</td><td><pre>name="Value 1" type="float"/></pre></td></holdingregister<>	address="8"	description="	Value"	<pre>name="Value 1" type="float"/></pre>
<holdingregister< td=""><td>address="9"</td><td>description="</td><td>dummy"</td><td><pre>name="dummy_9" type="float"/></pre></td></holdingregister<>	address="9"	description="	dummy"	<pre>name="dummy_9" type="float"/></pre>
<holdingregister< td=""><td>address="10"</td><td>description='</td><td>' dummy'</td><td><pre>' name="dummy_10" type="float"/></pre></td></holdingregister<>	address="10"	description='	' dummy'	<pre>' name="dummy_10" type="float"/></pre>
<holdingregister< td=""><td>address="11"</td><td>description=</td><td>' dummy'</td><td><pre>' name="dummy_11" type="float"/></pre></td></holdingregister<>	address="11"	description=	' dummy'	<pre>' name="dummy_11" type="float"/></pre>
<holdingregister< td=""><td>address="12"</td><td>description=</td><td>' dummy'</td><td><pre>' name="dummy_12" type="float"/></pre></td></holdingregister<>	address="12"	description=	' dummy'	<pre>' name="dummy_12" type="float"/></pre>
<holdingregister< td=""><td>address="13"</td><td>description='</td><td>' Value'</td><td>' name="Value 1" type="float"/></td></holdingregister<>	address="13"	description='	' Value'	' name=" Value 1 " type="float"/>
<holdingregister< td=""><td>address="14"</td><td>description='</td><td>' Value'</td><td>' name="Value 1" type="float"/></td></holdingregister<>	address="14"	description='	' Value'	' name=" Value 1 " type="float"/>
<holdingregister< td=""><td>address="15"</td><td>description='</td><td>' Value'</td><td>' name="Value 1" type="float"/></td></holdingregister<>	address="15"	description='	' Value'	' name=" Value 1 " type="float"/>

4.2 PROPERTIES

4.2.1 TCPENDPOINT

Property	Value Op-	Manda-	Example	Description
	tions	tory		
name	string	yes	"My Endpoint"	Friendly name of the Endpoint
host	string	yes	"192.168.1.100"	Hostname of the Endpoint, IP address or
				hostname allowed
port	number	yes	"502"	IP-Port of the Endpoint, default is 502
polling_rate	rate [0.1Hz	no	"1Hz"	Polling / reading rate of the endpoint
	- 10Hz]			
response_timeout	time	no	"0.25s"	Timeout for waiting on response of the
				endpoint

4.2.2 DEVICE

Property	Value	Manda-	Example	Description
	Options	tory		
name	string	yes	"My Device"	Friendly name of the Device
device_type	string	yes	"MyDevice-	Name of the used device, must be available as
			Туре"	ModbusDevice
unit_id	number	yes	"1"	Unit_Id of the device, typically "1"
	[0-255]			

4.2.3 MODBUSDEVICE

Property	Value Options	Mandatory	Example	Description
name	string	yes	"MyDeviceType"	Friendly name of the Modbus Device
byte_order	see byte	no	"big_endian"	Byte / Word Order
	order			

4.2.4 REGISTER

- Coil use function code 0x01 for reading.
- DiscreteInput use function code 0x02 for reading.
- HoldingRegister use function code 0x03 for reading.
- InputRegister use function code 0x04 for reading.

Property	Value Options	Manda-	Example	Description
		tory		
address	number [0-65535]	yes	"40000"	Register Start address, starting with 0
name	string	yes	"Value 1"	Channel name of the Modbus Register in OXYGEN
type	data_type	yes	"int16"	Data type of the register value
scale_mode	"const_linear" or "sunssf"	no	"const_lin- ear"	Scaling Mode
scale	number	no	"0.1"	Scaling factor, only valid if scale_mode == "const_linear"
offset	number	no	"100"	Scaling offset, only valid if scale_mode == "const_linear"
scale_reg	register	no	"1234"	Scaling register, only valid if scale_mode == "sunssf"
min	number	no	"-100"	Minimum display value range [RESERVED]
max	number	no	"100"	Maximum display value range [RESERVED]
unit	string	no	"V"	Value Unit
nan	value	no	"Oxffff"	Value to be treated as NaN
byte_order	see byte or- der	no	"big_endian"	Byte / Word Order
description	string	no	"Descrip- tion"	Channel description of the Modbus Regis- ter in OXYGEN

4.2.5 DATATYPES

-			
Name	Word Count	Description	Value Range
"uint16"	1	Unsigned Integer 16 Bit	0 to 32767
"int16"	1	Signed Integer 16 Bit	-16384 to 16383
"uint32"	2	Unsigned Integer 32 Bit	0 to 2^32-1
"int32"	2	Signed Integer 32 Bit	-2^31 to 2^31-1
"float"	2	IEE 754 Floating Point Single	+-3.402823e+38
"double"	4	IEE 754 Floating Point Double	+-1e+308

4.2.6 BYTE ORDER

Name	Synonym	Description
"abcd"	"big_endian"	Decode Data in Big Endian matter (High-Byte before Low-Byte)
"dcba"	"little_endian"	Decode Data in Little Endian matter (Low-Byte before High-Byte)
"cdab"	-	Decode Data in Mixed-Byte Order
"badc"	-	Decode Data in Mixed-Byte Order