Modbus/TCP

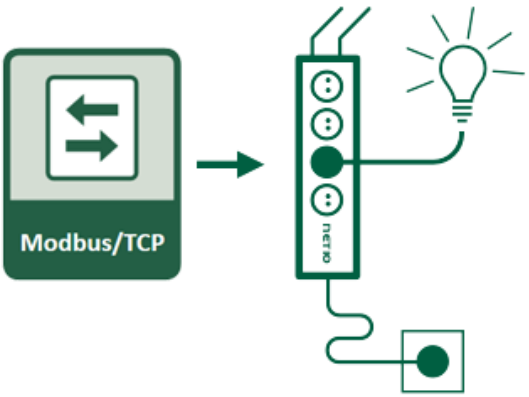
NETIO M2M API protocols docs

Protocol version: NETIO Modbus/TCP specification v12

Short summary

Modbus/TCP is a Modbus variant used for communications over TCP/IP networks, typically connecting over port 502. Modbus is a serial communications protocol originally published by Modicon (now Schneider Electric) in 1979 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/TCP is implemented in NETIO 4x products as M2M API protocol. NETIO device is Modbus/TCP server (slave) sharing output states and energy measured values on the Bit / Bytes registers.

NETIO power outputs can be controlled with the Modbus/TCP protocol. Power Outputs can be power sockets 230V (NETIO 4 / NETIO 4All) or power outlets IEC-320 110/230V for NETIO 4C.

- Modbus/TCP M2M API protocol must be enabled in the WEB configuration of the device.
- All NETIO 4x devices (NETIO 4 / 4All / 4C) supports controlling the Power outputs
- To increase security, there is the **IP filter** on the web interface. You can define IP range of allowed IP addresses.

Supported devices

- PowerCable Modbus 101x
- NETIO 4All
- NETIO 4 (Energy metering not supported)
- NETIO 4C (Energy metering not supported)

*Note: **NETIO 4x** means all NETIO 4 devices (NETIO 4 / 4All / 4C)*

Supported devices and firmware

NETIO 4x firmware – 3.1.0 and later

Modbus functions & properties

- **01** = Read Coils (NETIO: read On/Off state of one power output)
- **02** = Read Discrete Input
- **03** = Read Holding Registers (NETIO: read how many outputs available on the device)
- **04** = Read Input Registers (NETIO: read measurement values)
- **05** = Write Single Coil (NETIO: write On/Off state to one power output)
- **06** = Write Single Register (NETIO: write Action (Toggle for example) to one power output)
- **15** (0x0F) = Write Multiple Coils
- **16** (0x10) = Write Multiple Registers

Protocol properties

- Modbus/TCP **Unit ID** is ignored. Can be anything 0 to 247
- Output number is the same as written on the device (1 to 4), here used as xx = 01 to 04.
- **Wire Address** = **Register Address** – **1**
  - The **Wire Address** **starts from 0**
  - The **Register Address** starts from **1**
  - Register Address is called “PLC based addressing” in some applications (Modbuss Poll for example)
  - Some utilities using the Wire Address (QModMaster used in AN27 for example)

Quick start with Modbus/TCP & NETIO

Output (power socket) addressing

Output (power socket) number	**Wire Address** (“R” = register type address as described below)		**Register Address** (“R” = register type address as described below)	
(“x”)	Format (“Rxx”)	*Example*	**Format (“Rxx”)**	***Example***
1	R01	*101*	**R02**	***102***
2	R02	*102*	**R03**	***103***
3	R03	*103*	**R04**	***104***
4	R04	*104*	**R05**	***105***

How to control output as a Coil

Register Address	R / W	Type	Function	Description	Read/Write Values
1xx	R	bit	01	Read state of xx power sockets (0/False = off , 1/True = on)	0 or 1 False or True
1xx	W	bit	05	Set action to the xx output (0/False = off , 1/True = on)	0 or 1 False or True

Example

Register Address	Wire Address	R / W	Type	Function	Description	Read/Write Values
102	101	W	bit	05	Set 1th output to On	1 True
104	103	W	bit	05	Set 3th output to Off	0 False

- With function 05 write 1 (True) to Register address 102 to **switch Output 1 ON**
- With function 05 write 0 (False) to Register address 102 to **switch Output 1 OFF**
- With function 05 write 0 (False) to Register address 104 to **switch Output 3 OFF**

How to control output as a Holding register

Register Address	R / W	Type	Function	Description	Read/Write Values
1xx	R	uInt16	03	Read state of xx power socket output as 2 bytes value (0 or 1 as output Off/On)	0 or 1
1xx	W	uInt16	06	Set action to the xx output (0=off, 1=on, 2=short OFF, 3=short ON, 4=toggle, 5 = nothing)	0 to 5

Examples

- With function 03 Read **state of Output 1** from Register address 102 (Wire Address 101)
- With function 06 write 1 to Register address 102 to **switch Output 1 ON**
- With function 06 write 4 to Register address 102 to **Toggle Output 1**
- With function 06 write 4 to Register address 105 to **Toggle Output 4**

General NETIO 4x output functions

Output Actions – "write" function

- **0 –** Turn **OFF**
- **1 –** Turn **ON**
- **2 –** Short OFF delay (restart)
- **3 –** Short ON delay
- **4 –** Toggle (invert the state)
- **5 –** No change

Output Status – "read" function

- **0** – Power **OFF**
- **1** – Power **ON**

Short ON / OFF delay

This command switches a power output On / Off for a defined time. It is useful for example to power-cycle a server with a defined switch-off time, or to switch on a pump for a defined time.

This "short" delay is protected: the power output will remain in the defined state regardless of any other M2M requests received. During this time, the output state can only be changed by pressing the button on the NETIO device and this action cancel M2M short ON/OFF command for the particular output. Other requests to control the particular output are simply ignored.

The short ON / OFF delay interval can be defined in the device web administration. It is specified in ms (milliseconds) and rounded up to hundreds of milliseconds (0,1s).

This interval is unique for Modbus/TCP protocol. It is valid only for a single protocol session (the following short ON / Short OFF command). When the connection is closed or restarted, the interval is reset to the device default value (defined in the web administration for each output).

Security issues

Do not use default usernames and passwords! Keep your Ethernet and WiFi networks secured.

Power-Up outputs state

All outputs are Off during the first 25 to 30 seconds after power-up.
After this time, all outputs are set to the selected state:

- **Last Output state**
  After a power outage, the NETIO device sets each power output to the last stored state of this one output. The current state of each power output (socket/power outlet) is internally stored every 8 seconds.

Note: **Function Scheduler** is checked in Power-Up initialization. When enabled, it can affect one or more power output stated based on current time and date.

Custom based **Lua scripts** can affect output stated too.

Energy metering variables in NETIO 4All

Since NETIO fw 3.0.0 and later, there are 23 variables available for NETIO energy metering in NETIO 4All.

Parameters for **each power output**:

Variable	Unit	Description
4x Current	mA	Immediate current for the specific power output
4x TPF (True Power Factor)	-	Immediate True Power Factor for the specific power output
4x Load	W	Immediate load for the specific power output
4x Energy	Wh	Immediate Energy counter value for the specific power output

Parameters for the **whole NETIO 4All device**:

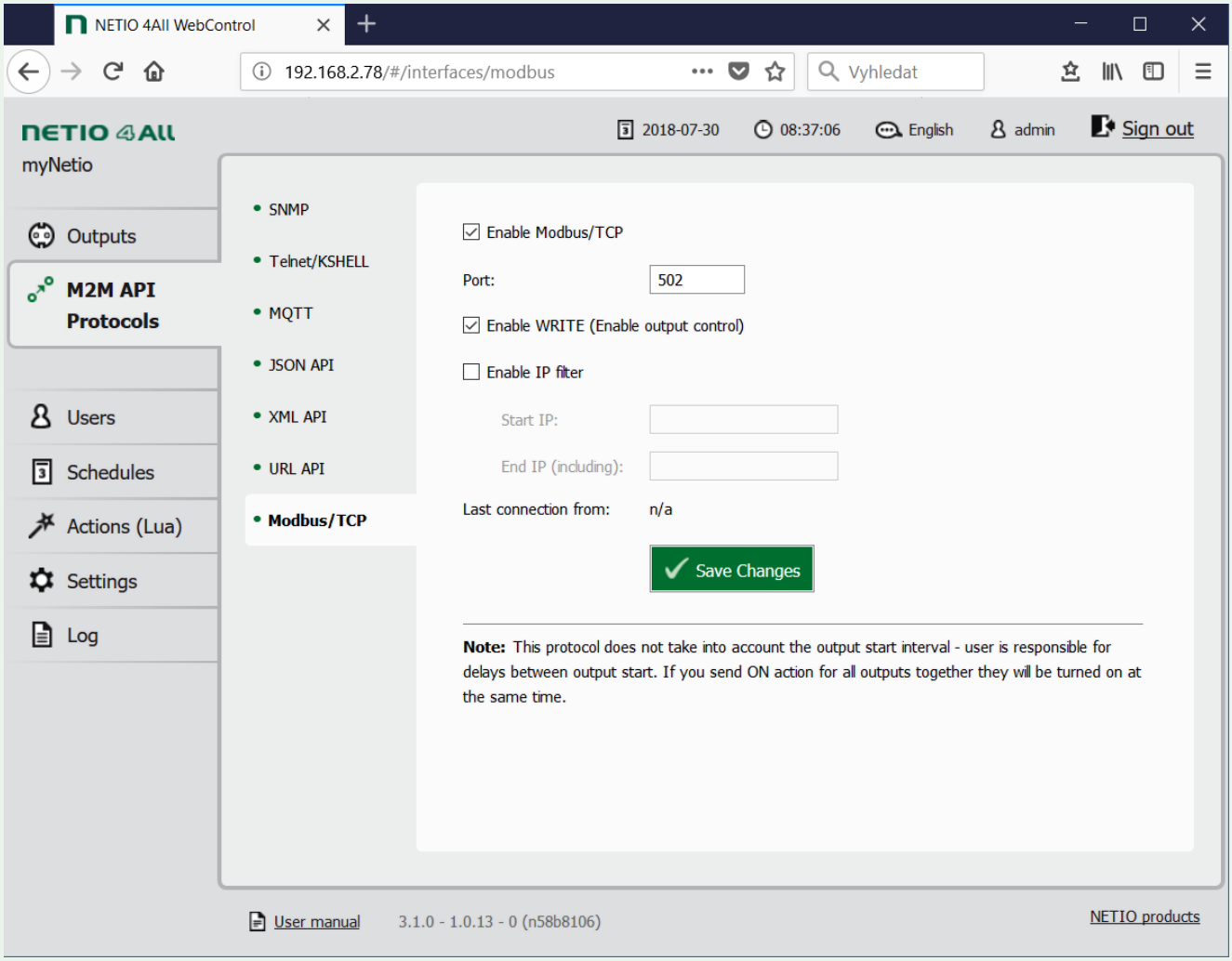
Variable	Unit	Description
1x Voltage	V	Immediate voltage
1x Frequency	Hz	Immediate frequency
1x TotalCurrent	mA	Immediate total current through all power outputs
1x Overall TPF (True Power Factor)	-	Immediate True Power Factor – weighted average from all meters
1x TotalLoad	W	Total Load of all power outputs (device's own internal consumption is not included)
1x TotalEnergy	Wh	Immediate value of the Total Energy counter
1x EnergyStart	-	Date and time of the last reset of all energy counters

Energy metering variables in NETIO PowerCable Modbus 101x:

Variable	Unit	Description
1x Voltage	V	Immediate voltage
1x Frequency	Hz	Immediate frequency
1x Current	mA	Immediate total current
1x TPF (True Power Factor)	-	Immediate True Power Factor
1x Load	W	Total Load of all power outputs
1x Energy	Wh	Immediate value of the Total Energy counter
1x EnergyStart	-	Date and time of the last reset of energy counter
1x PhaseAngle	°	Phase shift between Current & Voltage -180° to +180°

NETIO 4x WEB configuration

M2M API protocols can be enabled and configured only over the web administration – select “M2M API Protocols” in the left-hand side menu and then select the “Modbus/TCP” tab.



Picture 1 –M2M API Protocols / Modbus/TCP settings GUI

- **Enable Modbus/TCP** – Enable/disable M2M API protocol
- **Port** – socket where Modbus/TCP is responding
- **Enable WRITE (Enable output control)** – Enable write function thus output control. Only read – monitoring function if not checked.
- **Enable IP filter** – Apply basic security – only IP addresses from defined range are allowed to communicate over Modbus/TCP M2M API protocol
    - **Start IP** – Start IP of the “safe” range
    - **End IP (including)** – End IP of the “safe” range including this address
- **Last connection from** – The last IP address Modbus/TCP communication has come from

NETIO Modbus/TCP registers

int16 2B signed (integer)
uInt16 2B unsigned (word)

Output (power socket) addressing

Output (power socket) number	**Wire Address** ("R" = register type address as described below)		**Register Address** ("R" = register type address as described below)	
("x")	Format ("Rxx")	*Example*	**Format ("Rxx")**	***Example***
1	R01	*101*	**R02**	***102***
2	R02	*102*	**R03**	***103***
3	R03	*103*	**R04**	***104***
4	R04	*104*	**R05**	***105***

Holding registers – NETIO I/O Configuration

Register Address	**Wire Address**	**R / W**	**Type**	**Function**	**Description**	**Read Values**
1	0	R	uInt16	03	How many Digital Inputs are on the device	0
2	1	R	uInt16	03	How many Digital Outputs (Power Sockets/Outlets) are on the device	4
3	2	R	uInt16	03	How many Power Sockets/Outlets with measured outputs	0 or 4

Example: Amount of outputs on the device

Register Address	**Wire Address**	**R / W**	**Type**	**Function**	**Description**	**Read Values**
2	1	R	uInt16	03	How many Digital Outputs (Power Sockets/Outlets) are on the device	4

Coils – NETIO I/O control COILS block

It's possible to control each power socket output only as 0/1 bit with the coils via Modbus/TCP.

Register Address	R / W	Type	Function	Description	Read/Write Values
1xx	R	bit	01	Read state of xx power sockets (0/False = off , 1/True = on)	0 or 1 False or True
1xx	W	bit	05	Set action to the xx output (0/False = off , 1/True = on)	0 or 1 False or True

Example: Set output

Register Address	Wire Address	R / W	Type	Function	Description	Read/Write Values
102	101	W	bit	05	Set 1th output to On	1 True
104	103	W	bit	05	Set 3th output to Off	0 False

Holding registers – NETIO I/O control WORD block

We recommend to use the registers to control the NETIO 4x device.
It's possible to write not only 0/1 as with coil bit output, but even standard NETIO output actions like Toggle or Short On / Short Off.

Register Address	R / W	Type	Function	Description	Read/Write Values
1xx	R	uInt16	03	Read state of xx power socket output as 2 bytes value (0 or 1 as output Off/On)	0 or 1
1xx	W	uInt16	06	Set action to the xx output (0=off, 1=on, 2=short OFF, 3=short ON, 4=toggle, 5 = nothing)	0 to 5
2xx	R	uInt16	03	Read "short" delay time for xx power socket (short ON/OFF time). The value is in tenths of second so 95 means 9,5sec	0 to 2^{16}-1
2xx	W	uInt16	06	Write "short" delay time for xx power socket (short ON/OFF time). The value is in tenths of second so 95 means 9,5sec	0 to 2^{16}-1

Note: Wire Address = Register Address – 1

Examples

Register Address	Wire Address	R / W	Type	Function	Description	Read/Write Values
102	101	R	uInt16	03	Read state of 1st power socket as 2 bytes value (0 or 1 as output)	0 or 1
105	104	R	uInt16	03	Read state of 4th power socket as 2 bytes value (0 or 1 as output)	0 or 1
102	101	W	uInt16	06	Toggle ouput 1 (Power socket /outlet)	4
105	104	W	uInt16	06	Toggle ouput 4 (Power socket /outlet)	4
202	201	R	uInt16	03	Read "short" delay time of 1st power socket, which is for example 2000ms	20
205	204	R	uInt16	03	Read "short" delay time of 1st power socket, which is for example 50s	500

Input registers – NETIO measure block

Register Address	R / W	Type	Function	Description	Read Values
1	R	uInt16	04	Power grid frequency [Hz*100]	0 to 2^{16}-1
2	R	uInt16	04	Voltage [V*10] – RMS	0 to 2^{16}-1
3	R	int16	04	TPF (True Power Factor) [-], value = TruePowerFactor *1000	-2^{15} to 2^{15}-1

Note: Input registers are available on the NETIO 4All model only.
Other models return zero value for such reads.

Examples

Register Address	Wire Address	R / W	Type	Function	Description	Read Values
1	0	R	uInt16	04	Power grid frequency 50 Hz	5000
2	1	R	uInt16	04	Voltage [V*10] – RMS, which for example say 232 V	2320
3	2	R	int16	04	TPF (True Power factor) [-] , value = TruePowerFactor *1000, example 0.853	853

Input registers – NETIO energy block

Input registers are available on the NETIO 4All model only.
Other models (NETIO 4 / 4C) return zero value for such reads.

Register Address	R / W	Type	Function	Description	Read Values
1xx	R	uInt16	04	xx Output current [mA] – RMS (xx = 00 Wire Addr.. for whole device)	0 to 2^{16}-1
2xx	R	uInt16	04	xx Output power [W] – Instantaneous RMS (xx = 00 Wire Addr.. for whole device)	0 to 2^{16}-1
3(01+x*2)	R	uInt16	04	x Output energy counter [Wh] - 4B value, this are 2 upper bytes (x = 0 for whole device)	0 to 2^{16}-1
3(01+x*2+1)	R	uInt16	04	x Output energy counter [Wh] - 4B value, this are 2 lower bytes (x = 0 for whole device)	0 to 2^{16}-1
4xx (1)	R	Int16	04	xx Phase shift [°] (xx = 00 Wire Addr.. for whole device)	0 to 2^{16}-1

Note 1: Phase Shift measurement is supported in NETIO PowerCable Modbus only.

Examples

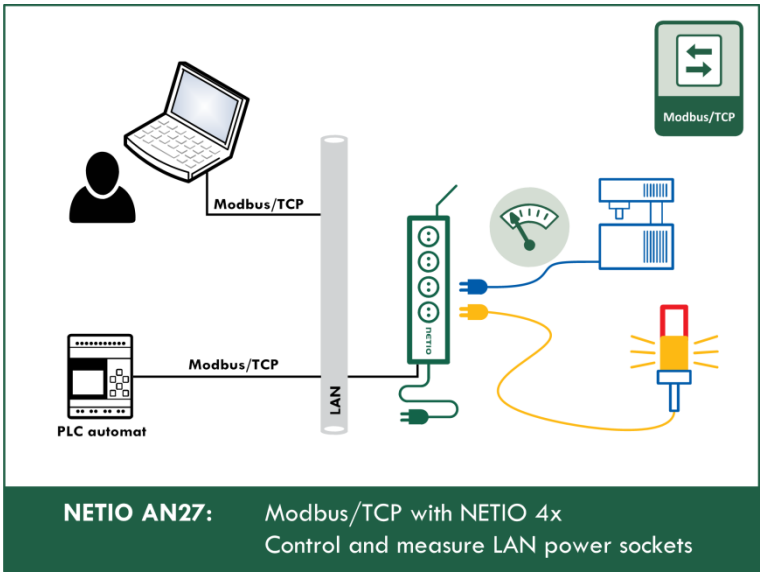
Register Address	Wire Address	R / W	Type	Function	Description	Read Values
102	101	R	uInt16	04	1st Output current 180 mA RMS	180
101	100	R	uInt16	04	Whole device (all outputs together) current 800 mA RMS	800
303	302	R	uInt16	04	1th Output energy counter 1135 Wh - 4B value, this are 2 upper bytes	0
304	303	R	uInt16	04	1th Output energy counter 1135 Wh - 4B value, this are 2 lower bytes	1135
309	308	R	uInt16	04	4th Output energy counter 950 Wh - 4B value, this are 2 upper bytes	0
310	309	R	uInt16	04	4th Output energy counter 950 Wh - 4B value, this are 2 lower bytes	950
301	300	R	uInt16	04	All Output energy counter 3092 Wh - 4B value, this are 2 upper bytes	0
302	301	R	uInt16	04	All Output energy counter 3092 Wh - 4B value, this are 2 lower bytes	3092

NETIO AN (Application Note)

AN27: Modbus/TCP with NETIO 4x – Control and measure LAN power sockets

The AN27 Application Note demonstrates how to control NETIO 4x smart sockets and PowerCable Modbus devices using the Modbus/TCP protocol.

Modbus/TCP is the de-facto industry standard for connecting sensors and devices to PLCs, SCADA systems and smart home systems. NETIO products can be controlled over Modbus/TCP as 1-bit outputs (“coils”). In addition, values can be measured and advanced control is possible using registers.



>> Read the AN27 on www.netio-products.com

Document history

Document Revision	Publication Date	Description
1.0	8.8.2018	Initial release for FW 3.1.0, spec v12
1.1	28.8.2018	Small mistakes in doc FW 3.1.1, spec v12
1.2	13.9.2018	Better description
1.3	4.10.2018	PowerCable Modbus as supported device added
1.4	5.11.2018	PhaseShift support included
1.5	30.11.2018	AN27 link included.