Est. USA 1981

www.AKCP.com

sensorProbe+ Modbus Manual & AKCP PMS

Introduction to Modbus

MODBUS RTU is a non-proprietary serial communications protocol that is widely used in the process control industry actuation. The hardware over which MODBUS RTU communications are performed is not defined by the protocol. MODBUS RTU is supported on RS-232, RS-422, RS-485, Ethernet and other electrical standards. The sensorProbe+ supports RS-485 and Ethernet.

The Modbus/RTU protocol defines how a “master” device polls one or more “slave” devices to read and write data in real time over RS-232, RS-422, or RS-485 serial data communication. A slave device cannot volunteer information; it must wait to be asked for it. The master will write data to a slave device’s registers, and read data from a slave device’s registers. A register address or register reference is always in the context of the slave’s registers.

Modbus/TCP, an extension of Modbus/RTU, defines how Modbus/RTU and Modbus/ASCII messages are encoded within and transported over TCP/IP-based networks. Modbus/TCP is just as simple to implement and flexible to apply as the original Modbus/RTU.

The sensorProbe+ can represent both “master” and “slave” devices and supports both Modbus RTU and Modbus TCP protocols.

Transactions on Modbus Networks

Controllers communicate using a master–slave technique, in which only one device (the master) can initiate transactions (called ‘queries’). The other devices (the slaves) respond by supplying the requested data to the master, or by taking the action requested in the query. Typical master devices include host processors and programming panels. Typical slaves include programmable controllers. The master can address individual slaves, or can initiate a broadcast message to all slaves. Slaves return a message (called a ‘response’) to queries that are addressed to them individually. Responses are not returned to broadcast queries from the master.

The Modbus protocol establishes the format for the master’s query by placing into it the device (or broadcast) address, a function code defining the requested action, any data to be sent, and an error–checking field. The slave’s response message is also constructed using Modbus protocol. It contains fields confirming the action taken, any data to be returned, and an error–checking field. If an error occurred in receipt of the message, or if the slave is unable to perform the requested action, the slave will construct an error message and send it as its response.
The slave must respond only if its address is recognized, and must respond within a certain time period or the master will call it a "no response" error.

Modbus Limitations

- Since Modbus was designed in the late 1970s to communicate to programmable logic controllers, the number of data types is limited to those understood by PLCs at the time. Large binary objects are not supported.
- No standard way exists for a node to find the description of a data object, for example, to determine if a register value represents a temperature between 30 and 175 degrees.
- Since Modbus is a master/slave protocol, there is no way for a field device to "report by exception" (except over Ethernet TCP/IP, called open-mbus)- the master node must routinely poll each field device, and look for changes in the data. This consumes bandwidth and network time in applications where bandwidth may be expensive, such as over a low-bit-rate radio link.
- Modbus is restricted to addressing 254 devices on one data link, which limits the number of field devices that may be connected to a master station (once again Ethernet TCP/IP being an exception).
- Modbus transmissions must be contiguous which limits the types of remote communications devices to those that can buffer data to avoid gaps in the transmission.
- Modbus protocol itself provides no security against unauthorized commands or interception of data.

Important notes for understanding Master/Client

Master = Client: Modbus specifications use the term client, but we simplified this to Master on the SP+ webUI.
Slave = Server: Modbus specifications use the term server, but we simplified this to Slave on the SP+ webUI.

We use the terms Master/Slave on the unit’s WebUI as it is more understandable for common users than Client/Server.

<u>VERY IMPORTANT NOTE:</u>

For the AKCP PMS (Power Monitoring Sensor) please refer to the information beginning on page # 24 of this manual.

sensorProbe+ support

SP2+ Expansion only supports Modbus with RJ45 connector (RTU and TCP) on its Expansion port.
When you use Modbus, you can’t connect expansion boards to the unit!
See the wiring diagram below about making a Modbus cable for the RJ45 connector.

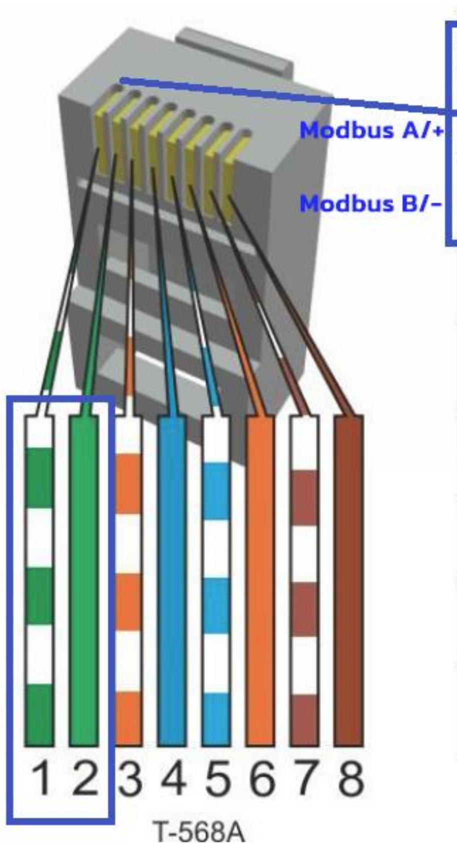
SPX+ also supports Modbus with the RJ45 Expansion port, and by adding an optional module, your unit can be equipped with 2x RS485 connectors. With this module the SPX+ unit can function as a Modbus Master and Slave device at the same time.
Every SP+ device supports up to 60 Modbus virtual sensors (4 devices, 15 sensors each).

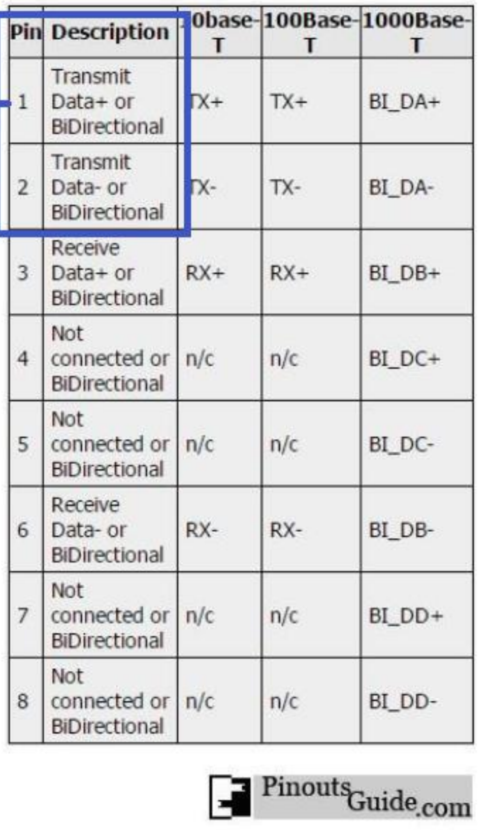
Note: some of the Web UI screenshots in this manual is from an earlier firmware, but the method of configuration hasn’t been changed since.

RJ45 Modbus cable wiring diagram

RS485 wiring on the RJ45 Expansion connector for the sensorProbe+ family.

Note: For just Modbus, only the pin 1 and 2 are used, being respectively, Modbus A/+ and Modbus B/-. You cannot use other sensor ports for Modbus other than the Expansion port.





Pin	Description	0base-T	100Base-T	1000Base-T
1	Transmit Data+ or BiDirectional	TX+	TX+	BI_DA+
2	Transmit Data- or BiDirectional	TX-	TX-	BI_DA-
3	Receive Data+ or BiDirectional	RX+	RX+	BI_DB+
4	Not connected or BiDirectional	n/c	n/c	BI_DC+
5	Not connected or BiDirectional	n/c	n/c	BI_DC-
6	Receive Data- or BiDirectional	RX-	RX-	BI_DB-
7	Not connected or BiDirectional	n/c	n/c	BI_DD+
8	Not connected or BiDirectional	n/c	n/c	BI_DD-

PinoutsGuide.com

1	RS485 TXD+	Modbus A/+
2	RS485 TXD-	Modbus B/-
3	RS485 RXD+	
4	Reset	
5	Unused	
6	RS485 RXD-	
7	Autosense	
8	Ground	

Licensing information

Modbus relies on existing licensing: virtual sensors and 3rd party PMS.
If you buy a Modbus RTU module (still under development) for the SPX+, you will get 10 additional free virtual sensors.

There are 5 Modbus features on the SP+ family and their licensing is as follows:

1. Virtual Sensors:
One Virtual Sensor can poll only 1 Modbus ID device and 1 register (TCP or RTU).
There are 5 free virtual sensors per unit and we charge 15$ per 5 additional Virtual Sensors.
SP+ supports a total of 60 Modbus virtual sensors (4 devices, 15 sensors each).

2. Modbus RTU PMS:
We have UI specific to setting up a PMS (like on our securityProbe series), and each unit can support up to 4 PMS. Our AKCP PMS are free to use.

3. 3rd Party PMS:
You can also upload a configuration file to use a 3rd party PMS in the PMS UI (see below).
This makes the job simpler to online the sensors and to manage the sensors, instead of using virtual sensors.
This feature is licensed separately.

4. Modbus notification:
One action sends one Modbus command (TCP or RTU), free to use.

5. SP+ Slave Modbus:
The SP+ can work as a Modbus Slave where onlined sensor values can be polled via Modbus protocol (TCP or RTU). You can read about how to configure the Modbus device ID for each sensor in this manual at the *Modbus Sensors Addressing* section. This feature is also free to use.

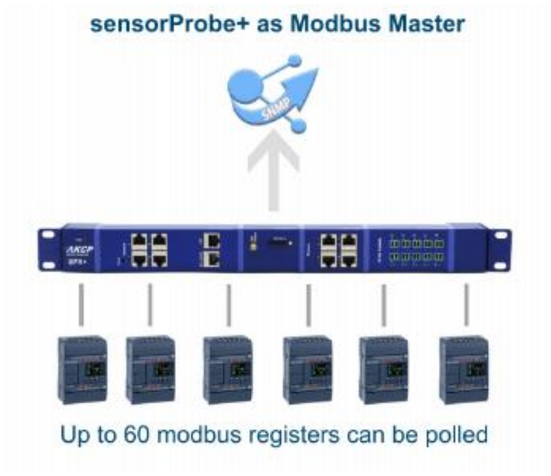
AKCP PMS (Power Monitor Sensor)

As noted above in item #2 the **AKCP PMS** can be used without any license. The AKCP PMS is setup using the Modbus Device setup and **NOT** using the virtual sensors. Again, please refer to the steps beginning on page #24 for the setup of the AKCP PMS.

SP+ Modbus concept diagrams

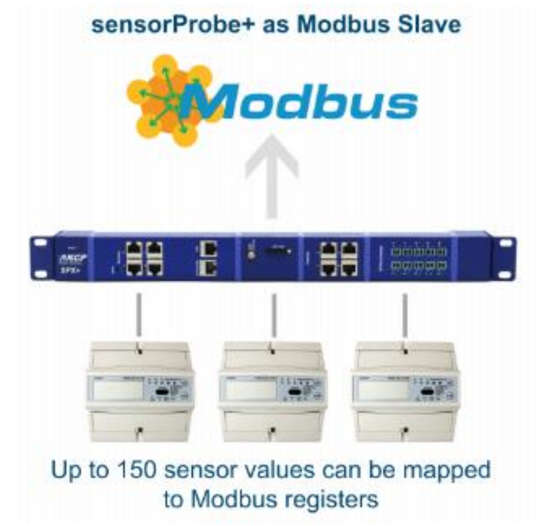
The sensorProbe+ series is a cost effective and robust solution that can act as a Modbus RTU <-> TCP/IP as well as Modbus RTU or TCP/IP <-> SNMP gateway. It can function as either Master or Slave.

Master
(Modbus Client mode)


sensorProbe+ as Modbus Master
SNMP
Up to 60 modbus registers can be polled

In Master mode, you can poll Slave devices via Modbus device IDs over Modbus RTU or TCP/IP.

Slave
(Modbus Server mode)


sensorProbe+ as Modbus Slave
Modbus
Up to 150 sensor values can be mapped to Modbus registers

In Slave mode, the SP+ will act as a Modbus RTU or TCP/IP <-> SNMP gateway and you can poll sensor data from another Modbus Master device.

Specifications

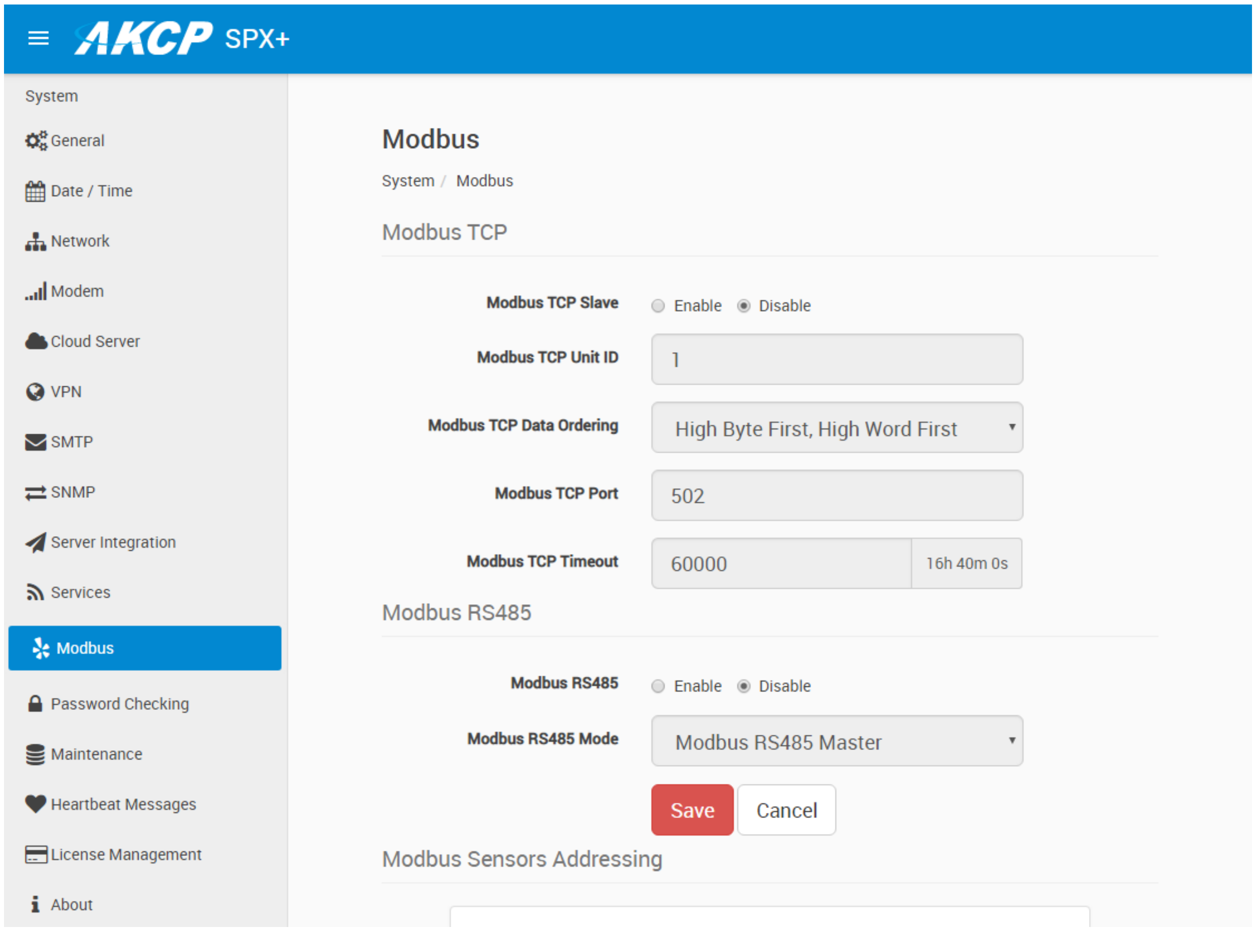
Modbus RTU	Modbus RTU connects through the Expansion port (EXP) as a 2-wire connection.
Speed	9600, 19200, 38400, 57600 and 115200
Parity	None, Even, Odd
Stop Bit	1, 2
Databit	8
Modbus TCP/IP	
Host	IP Address
Port	502, 1024-65535
Modbus Slave	
	Support Modbus Slave RTU and TCP/IP
	Can poll up to 60 values with Virtual sensors or up to 4 PMS.
Functions	0x01 - Read Coil Status
	0x02 - Read Input Status
	0x03 - Read Holding Registers
	0x04 - Read Input Registers

Data Type	16 bits unsigned int
	16 bits signed int
	32 bits unsigned int
	32 bits signed int
	32 bits IEEE floating point
Data ordering	low byte first, low word first
	low byte first, high word first
	high byte first, high word first
	high byte first, low word first
AKCP's Power Meter Sensor (PMS)	The SPX+ and SP2+ include a specific interface to online up to 4 AKCP's Power Meters via Modbus RTU
	3rd Party Power Meters support is available but requires the purchase of extra licences.
Modbus Notifications	The SPX+ and SP2+ support Modbus Write commands via the Notification feature
Functions	0x05 - Write Single Coil
	0x06 - Write Single Register
	0x0F - Write Multiple Coils (32 bits maximum)
	0x10 - write Multiple Registers (2 registers maximum)

Modbus Master	
	Support Modbus Master RTU and TCP/IP
Functions	0x03 - Read Holding Registers : 32 bits IEEE floating point value
	0x04 - Read Input Registers : 32 bits signed int value
Register Address List	All sensors monitored by the SPX+ or SP2+ have a Modbus register automatically assigned, and the user can edit the Modbus register assigned to the sensors.
	The register list can be consulted on the unit's WebUI page, in System >> Modbus

Setting up MODBUS on the sensorProbe+

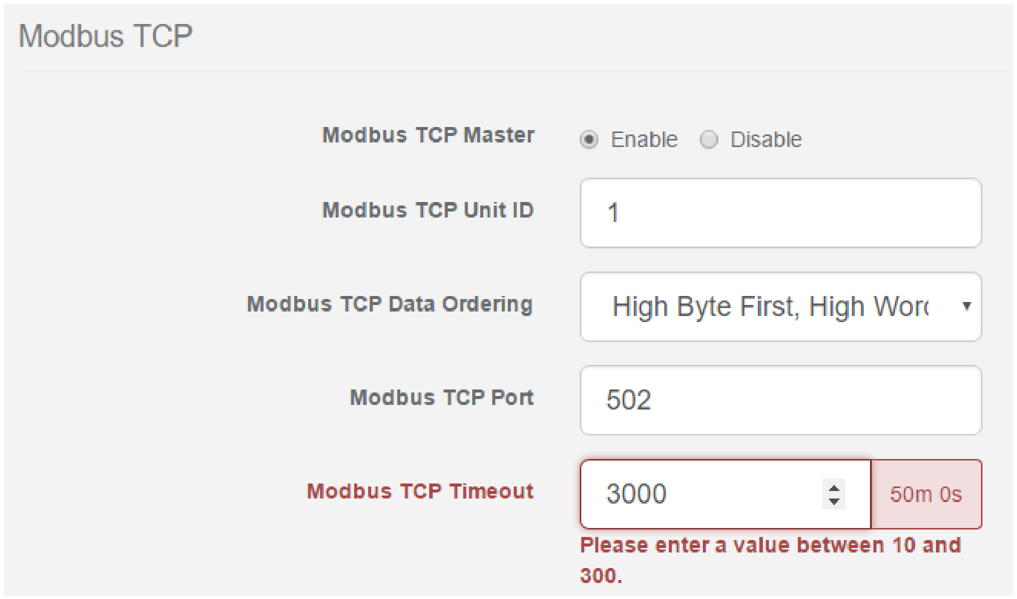
Note: some of the Web UI screenshots in this manual is from an earlier firmware, but the method of configuration hasn't been changed since.



Modbus TCP Slave	Enable [✓ Disable]
Modbus TCP Unit ID	1
Modbus TCP Data Ordering	High Byte First, High Word First
Modbus TCP Port	502
Modbus TCP Timeout	60000 (16h 40m 0s)
Modbus RS485	Enable [✓ Disable]
Modbus RS485 Mode	Modbus RS485 Master

From the "System" menu in the sensorProbe+ web interface, click on "Modbus" in the left hand column.

TCP Master
(Modbus Client mode)


Modbus TCP

Modbus TCP Master: (● Enable) ○ Disable

Modbus TCP Unit ID: 1

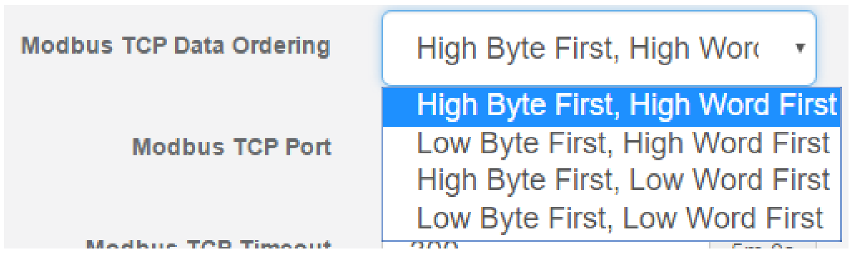
Modbus TCP Data Ordering: High Byte First, High Word

Modbus TCP Port: 502

Modbus TCP Timeout: 3000 — 50m 0s

Please enter a value between 10 and 300.

For setting up the Modbus TCP Master, first set the TCP port and TCP timeout (default TCP port is 502 and TCP timeout is 10 seconds). See screen shot above.


Modbus TCP Data Ordering: High Byte First, High Word

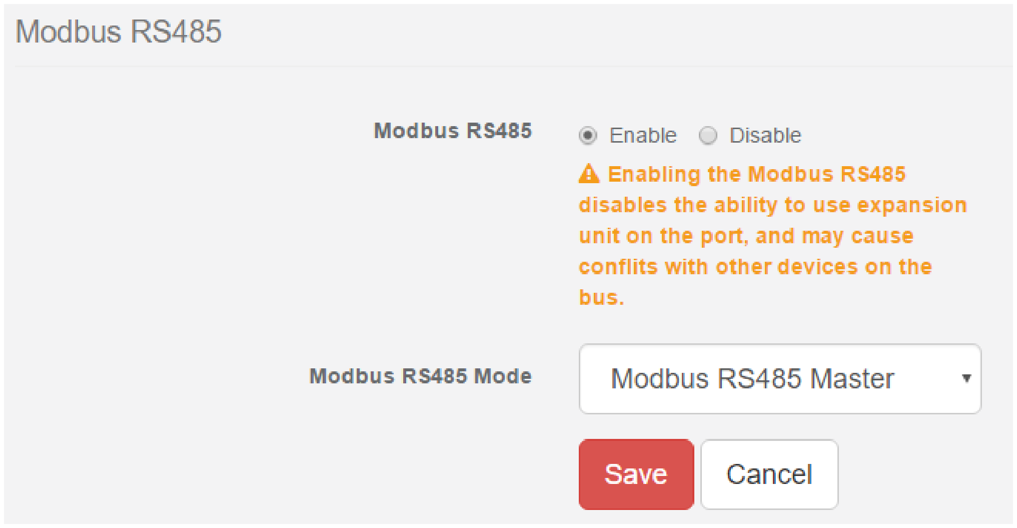
- High Byte First, High Word First
- Low Byte First, High Word First
- High Byte First, Low Word First
- Low Byte First, Low Word First

Modbus TCP Port

Additionally you have the option to define the data ordering from the drop-down list.

Note: You'll need to reboot the device after enabling/changing any Modbus setting.

RS485 Master
(Modbus Client mode)



Modbus RS485

Modbus RS485: [✓ Enable] Disable

⚠ Enabling the Modbus RS485 disables the ability to use expansion unit on the port, and may cause conflits with other devices on the bus.

Modbus RS485 Mode: Modbus RS485 Master

For setting up the Modbus RTU Master:

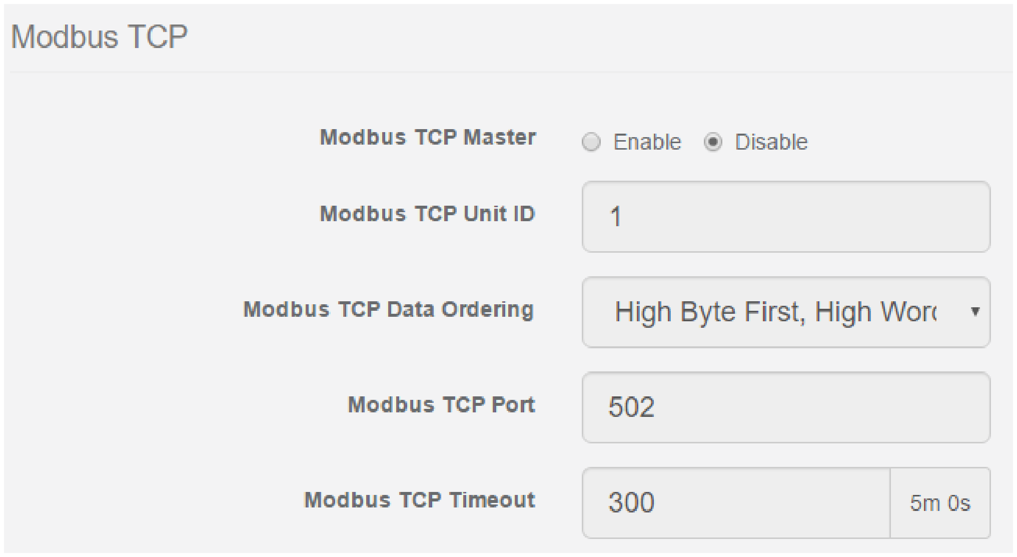
Enable the RS485 setting and choose the Modbus RS485 Master mode, then click **Save**.

Important: Please note the warning message when enabling RS485 support:
You won't be able to use the Expansion port, as it will be reserved for the Modbus signaling.

Note: You'll need to reboot the device after enabling/changing any Modbus setting.

TCP Slave

(Modbus Server mode)



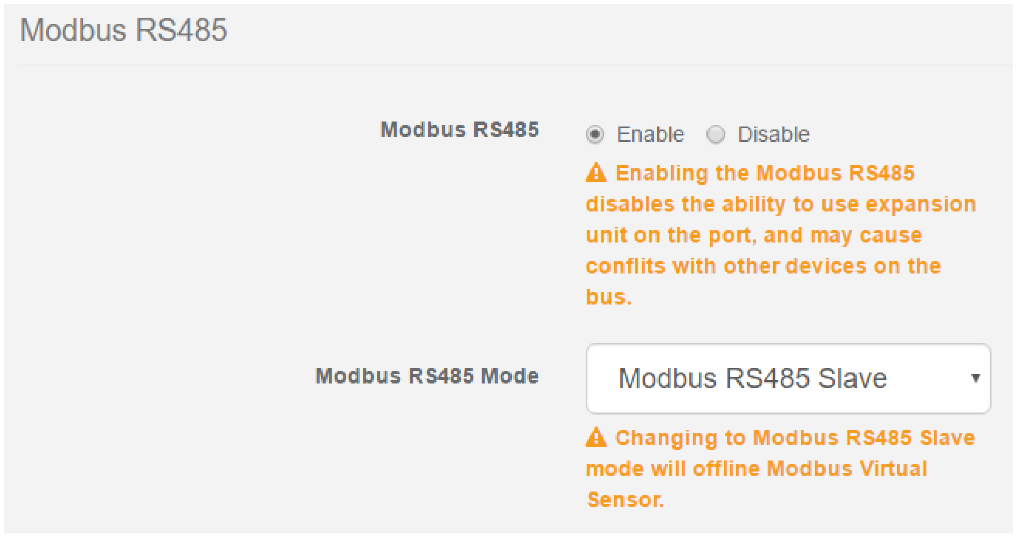
Modbus TCP

Modbus TCP Master	Enable [✓ Disable]
Modbus TCP Unit ID	1
Modbus TCP Data Ordering	High Byte First, High Word
Modbus TCP Port	502
Modbus TCP Timeout	300 5m 0s

For Modbus TCP Slave, there's no special settings required. Just ensure that the Modbus TCP Master option is on "disable" (default setting).

To be able to poll SNMP sensor values via Modbus, you'll need to assign Modbus device IDs to each sensor. See below for Modbus Sensors Addressing about how to set this up.

RS485 Slave
(Modbus Server mode)


Modbus RS485

Modbus RS485: [✓ Enable] Disable

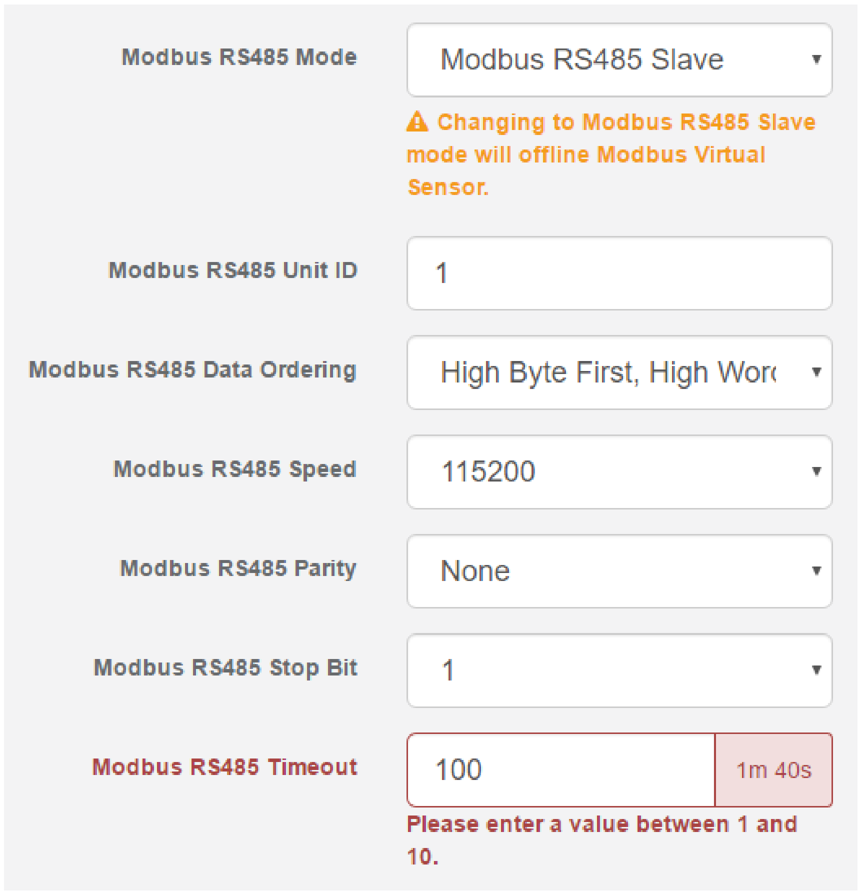
⚠ Enabling the Modbus RS485 disables the ability to use expansion unit on the port, and may cause conflits with other devices on the bus.

Modbus RS485 Mode: Modbus RS485 Slave

⚠ Changing to Modbus RS485 Slave mode will offline Modbus Virtual Sensor.

If you use Modbus RTU (this depends on your connection type) first set it to “enable” and for the Mode, choose the “Modbus RS485 Slave” from the drop-down list.

To be able to poll SNMP sensor values via Modbus, you’ll need to assign Modbus device IDs to each sensor. See below for Modbus Sensors Addressing about how to set this up.



Modbus RS485 Mode: Modbus RS485 Slave

⚠ Changing to Modbus RS485 Slave mode will offline Modbus Virtual Sensor.

Modbus RS485 Unit ID: 1

Modbus RS485 Data Ordering: High Byte First, High Worc

Modbus RS485 Speed: 115200

Modbus RS485 Parity: None

Modbus RS485 Stop Bit: 1

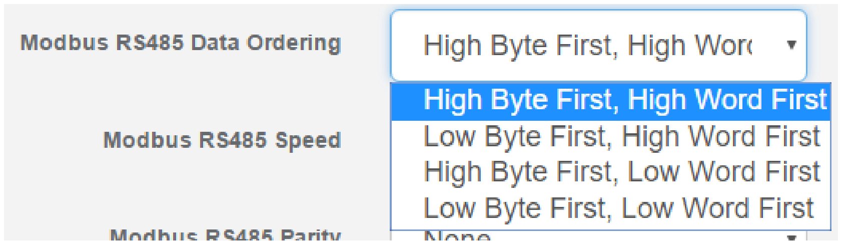
Modbus RS485 Timeout: 100 | 1m 40s

Please enter a value between 1 and 10.

You'll need to enter further settings for the Slave:

- Set the Modbus Address (Unit ID),
- Speed,
- Parity,
- Stop bits,
- Timeout

(The default Modbus Address is 1, Port Speed is 115200, Port Parity is None and Port Stop bit is 1)



Modbus RS485 Data Ordering: High Byte First, High Worc

- High Byte First, High Word First
- Low Byte First, High Word First
- High Byte First, Low Word First
- Low Byte First, Low Word First

Modbus RS485 Speed

Modbus RS485 Parity: None

Additionally you have the option to define the data ordering from the drop-down list.

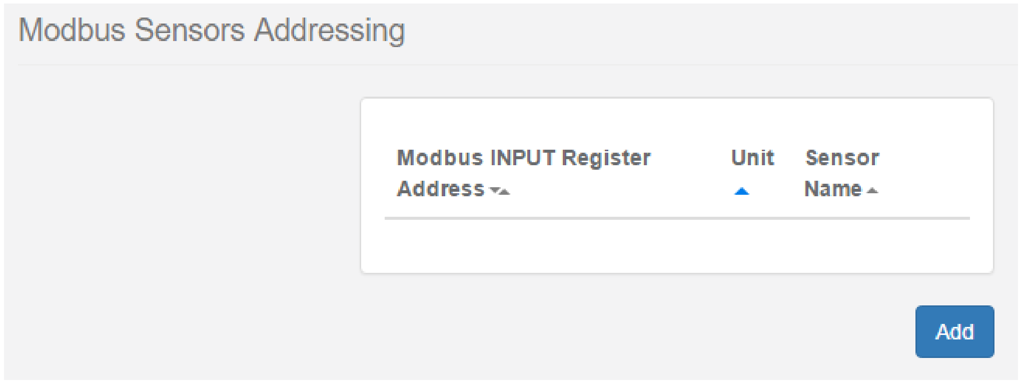
Important: Please note the warning messages when enabling RS485 support:

a) You won't be able to use the Expansion port, as it will be reserved for the Modbus signaling
b) If you enable the RS485 Slave and had any Modbus Virtual Sensors, these will become offline

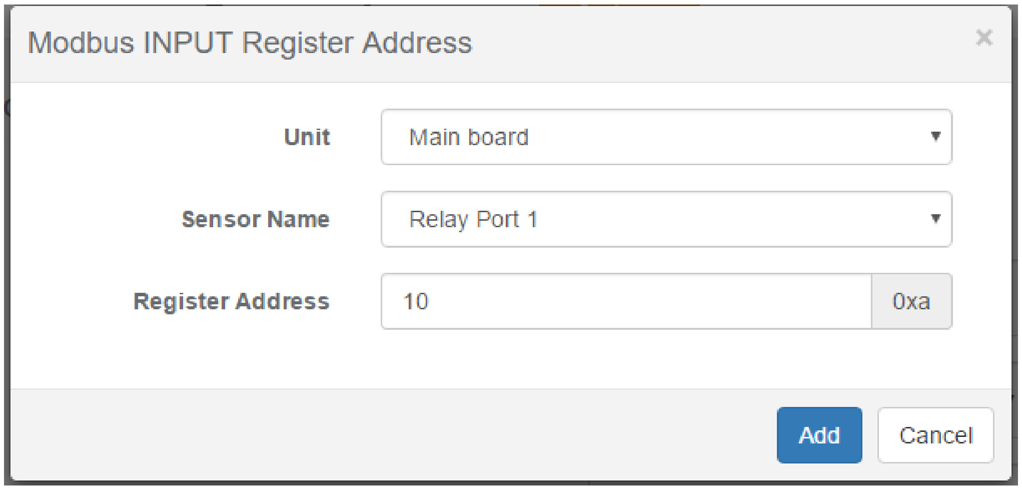
Note: You'll need to reboot the device after enabling/changing any Modbus setting.

Modbus Sensors Addressing

With the Modbus Sensors Addressing you can define the Modbus INPUT Register Address for each sensor that is online on the device.

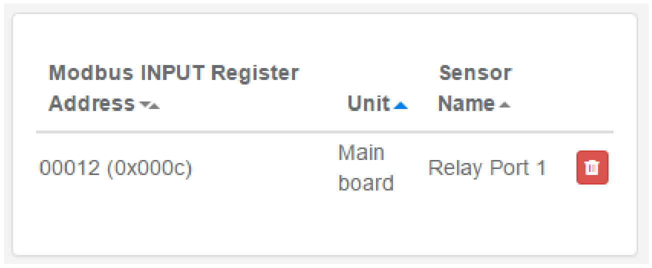


Click on the **Add** button, a popup window will open.


Modbus INPUT Register Address

Unit	Main board
Sensor Name	Relay Port 1
Register Address	10 0xa

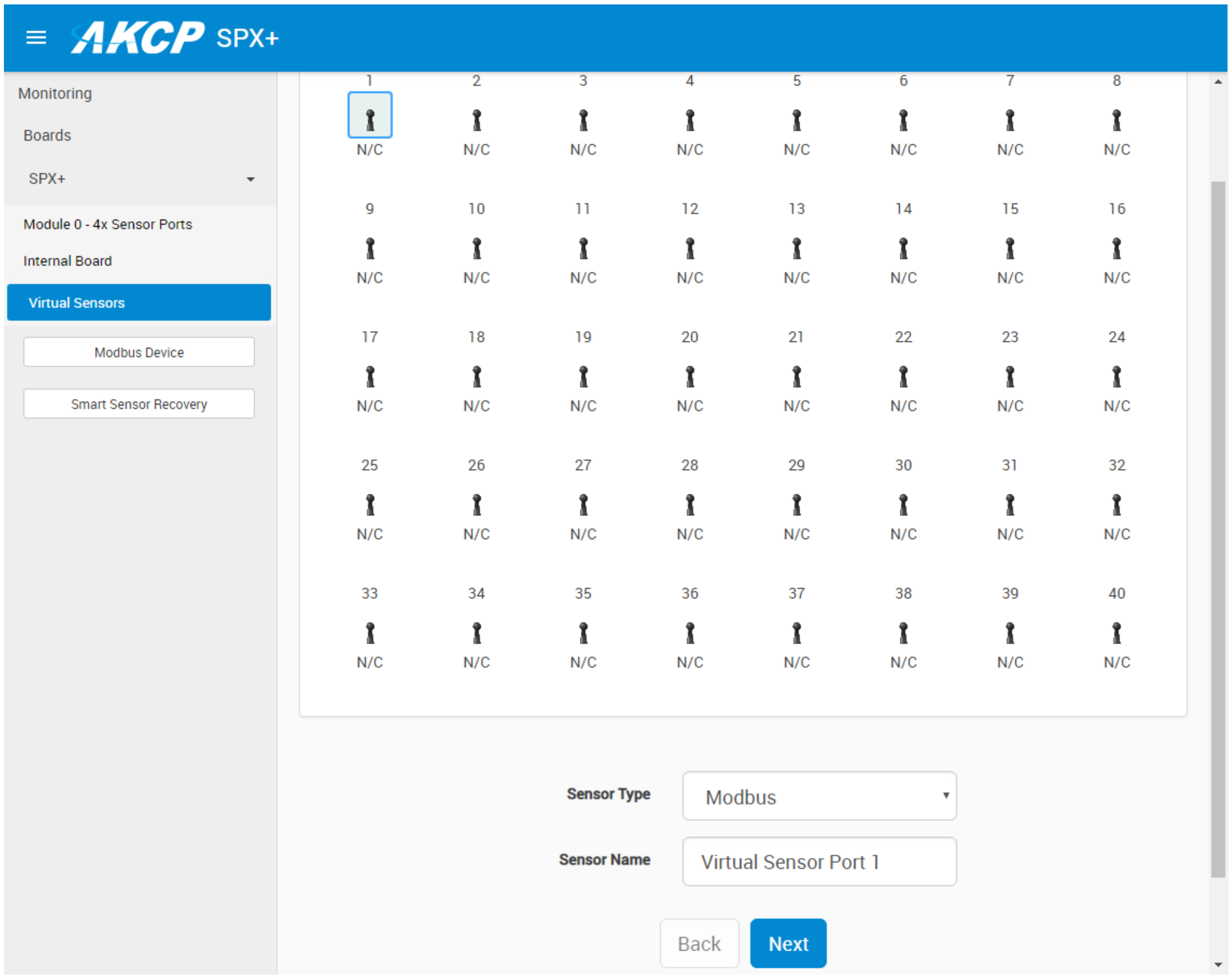
Select the Unit where the sensor is located, and select the sensor from the drop-down list.
Finally assign a Register Address to the given sensor.



Modbus INPUT Register Address	Unit	Sensor Name
00012 (0x000c)	Main board	Relay Port 1

Example: The Modbus INPUT Register Address of the Relay on port 1 is 12 (0x000c).

Modbus Virtual Sensor configuration

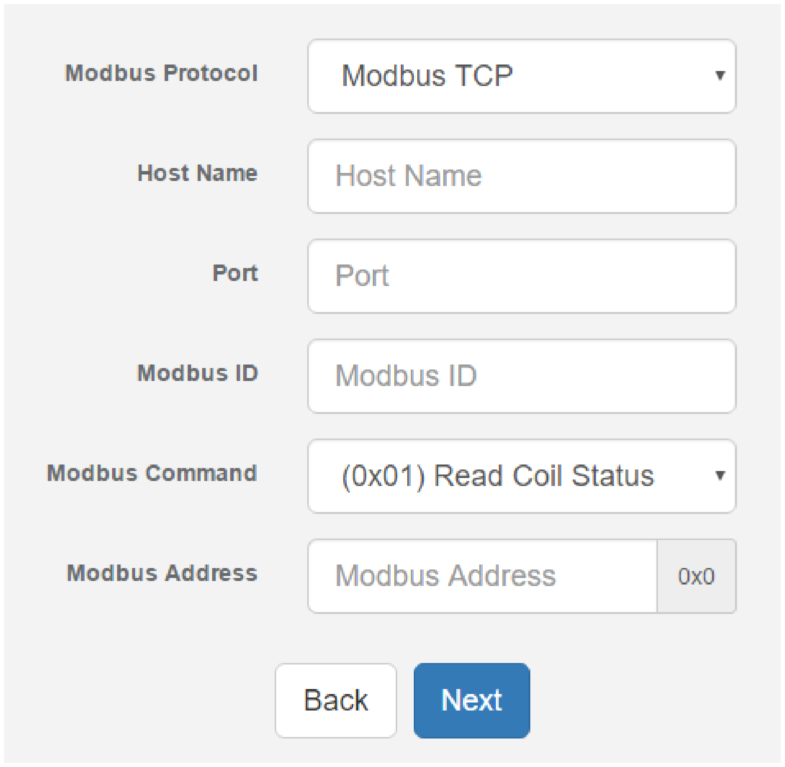


Sensor Type: Modbus
Sensor Name: Virtual Sensor Port 1

From the “Sensors” menu (on Master device) select Virtual Sensors, then set the “Sensor Type” to Modbus and then click “Next”. See screen shot above.

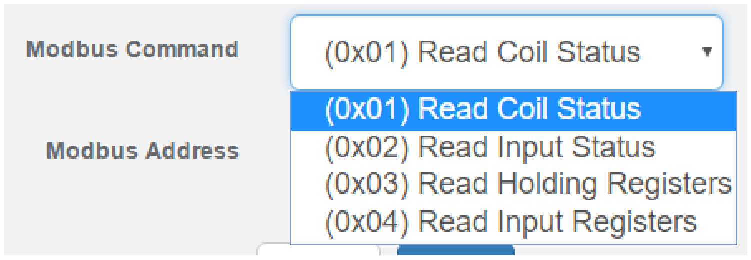
Note: You cannot use the Virtual Sensors in RS485 Slave mode.

For the Modbus TCP:

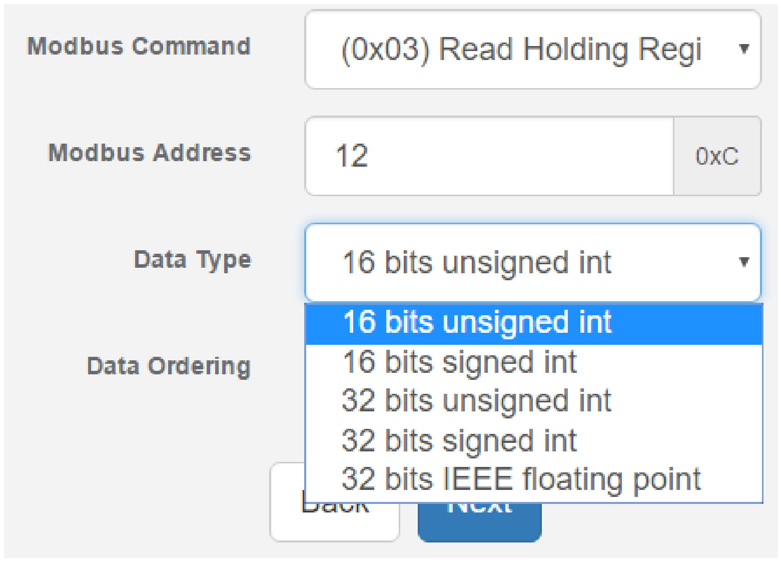


Modbus Protocol	Modbus TCP
Host Name	Host Name
Port	Port
Modbus ID	Modbus ID
Modbus Command	(0x01) Read Coil Status
Modbus Address	Modbus Address 0x0

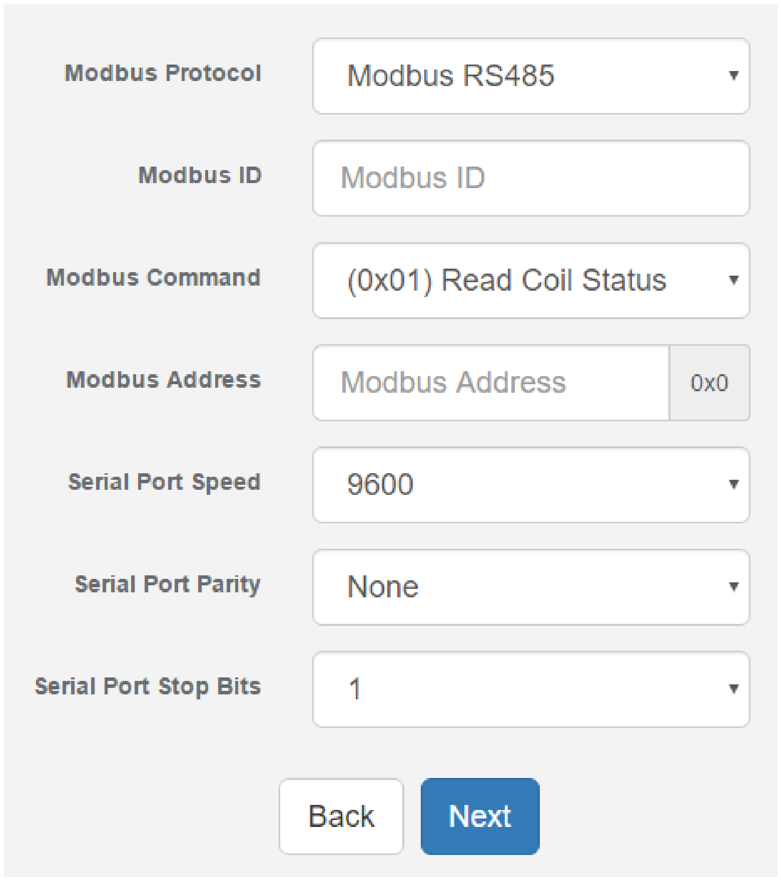
- Set the “Modbus Protocol” to Modbus TCP
- Enter the Slave device IP address/name in “Host Name”
- Enter the TCP port in “Modbus TCP Port”
- Enter the Modbus ID of the Slave
- Set the “Modbus command” to get the sensor value or get the sensor status:
  0x03 Read Holding Registers for getting the sensor status
  0x04 Read Input Registers for getting the sensor value
- Now set the Modbus INPUT Register Address value to Modbus Address (see the section above for the Modbus INPUT Register Address Value)
  Example; Set 12 for the read value of the Relay on port 1. See screen shot above



If you use Read Holding Registers or Read Input Registers, you'll have some additional options available to select the Data Type and Data Ordering:


Modbus Command: (0x03) Read Holding Regi
Modbus Address: 12 0xC
Data Type: 16 bits unsigned int
Data Ordering

For Modbus RTU:

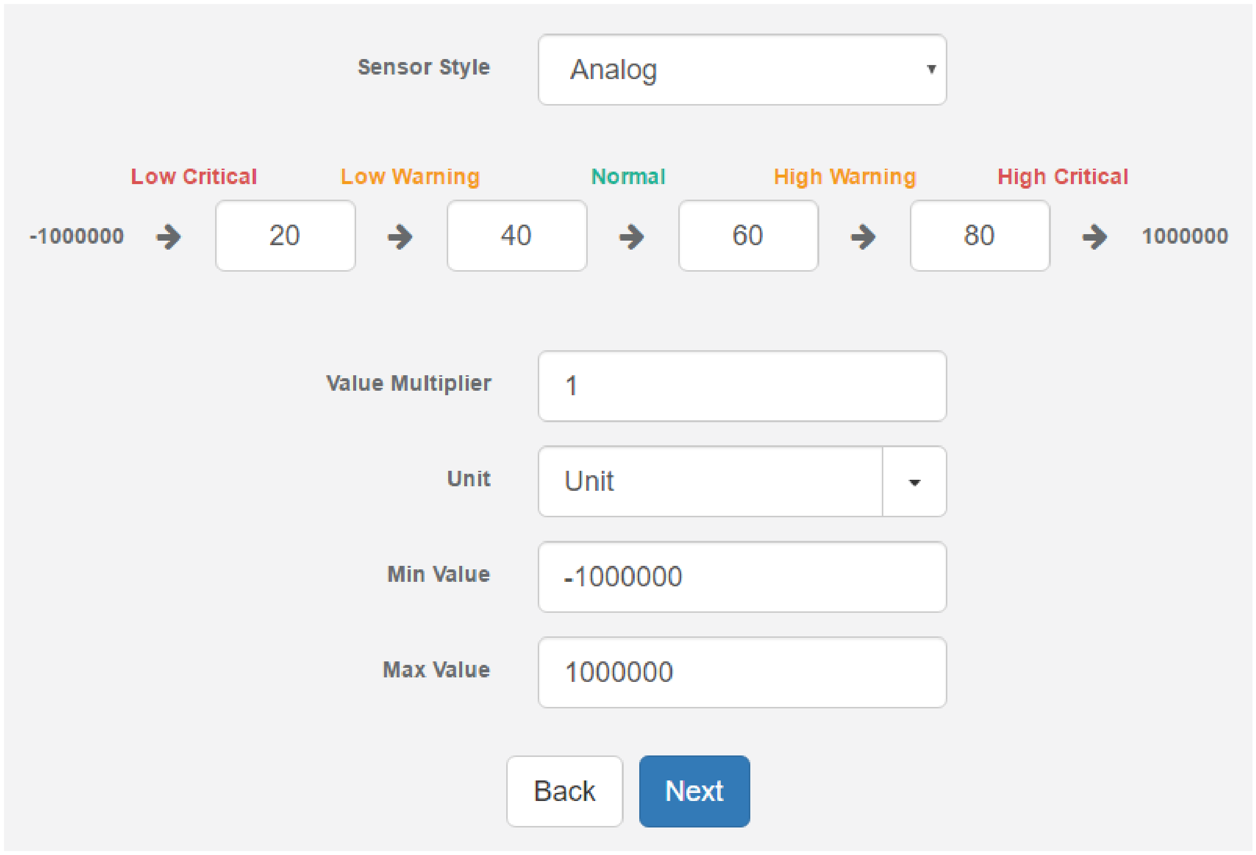


Modbus Protocol	Modbus RS485
Modbus ID	Modbus ID
Modbus Command	(0x01) Read Coil Status
Modbus Address	Modbus Address 0x0
Serial Port Speed	9600
Serial Port Parity	None
Serial Port Stop Bits	1

- Set “Modbus Protocol” to Modbus RS485
- Enter the Slave device’s ID in “Modbus ID”
- Select the Modbus command for getting the sensor value or the sensor status (0x03 Read Holding Register for getting the sensor status and 0x04 Read Input Register for getting the sensor value).
- Input the same “Modbus Address” value of the Modbus Slave ID
- Set “Serial Port Speed” (this setting should be the same as the Slave)
- Set “Serial Port Parity” (this setting should be the same as the Slave)
- Set “Serial Port Stop Bits” (this setting should be the same as the Slave)
- Input the Modbus INPUT Register Address value into “Modbus Register Address”. Example; Set 12 for the read value of the Relay on port 1.

Sensor Style

Analog Style


Sensor Style: Analog

Low Critical	Low Warning	Normal	High Warning	High Critical
-1000000 → 20	20 → 40	40 → 60	60 → 80	80 → 1000000

Value Multiplier: 1

Unit: Unit

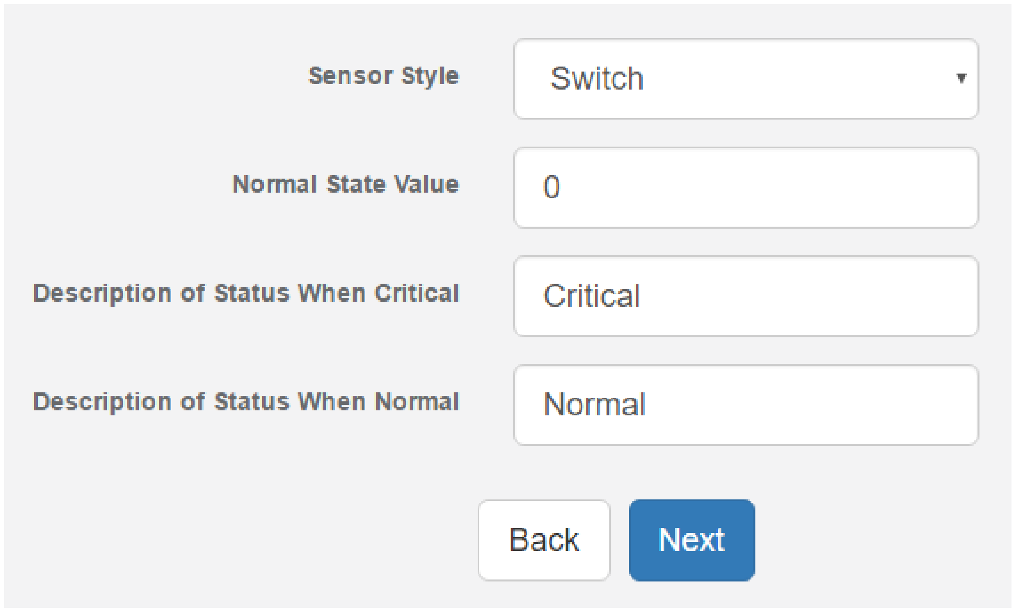
Min Value: -1000000

Max Value: 1000000

“Value Multiplier”, Set this for multiple the reading with normal value (Example; if raw value is 1234 and needs to show a value to 12.34, then this should be set to x0.01. Default is x1).
Input the Unit of sensor value into the “Unit Text” field, or click the drop-down list next to the input box to input a special character.

Input the Minimum and Maximum of the sensor values, set the Low Critical, Low Warning, High Warning and High Critical thresholds as shown in the screen shot above, then click “Next”.

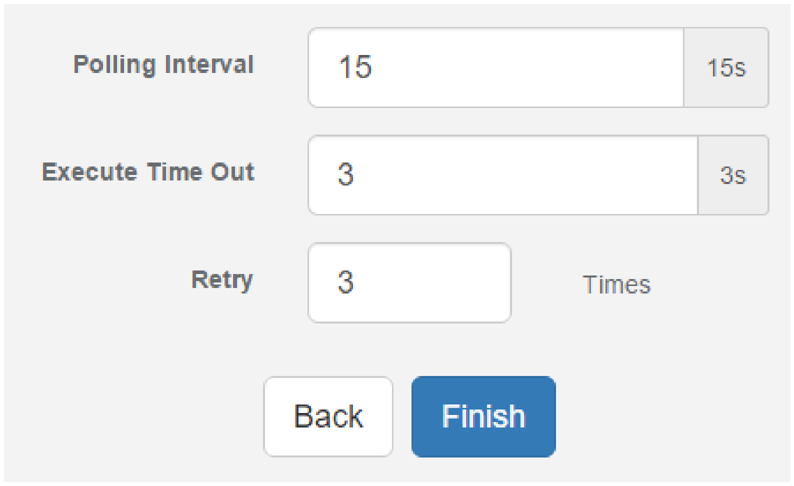
Switch Style



Sensor Style	Switch
Normal State Value	0
Description of Status When Critical	Critical
Description of Status When Normal	Normal

Input the Normal value of the sensor into “Normal State Value”.
Input the Description of Normal and Critical status into “Description of Status When Normal” and “Description of Status When Critical”.

Set Time to Polling



Polling Interval	15	15s
Execute Time Out	3	3s
Retry	3	Times

- Input the time to polling into “Polling Interval” (Default is 15 secs)
- Input “Execute Time Out” (Default is 10 secs) and “Retry” (Default is 3 times)
- Click the “Finish” button

AKCP Power Meter support & CT’s

The AKCP Power Monitoring Sensor (PMS) is supported on the sensorProbe+ family through Modbus monitoring.
The PMS is used to monitor the amount of the following values: Phase Line Voltages, Phase Line Current, Watts, Power Factor and Kwatt or Kilowatt hours.

Each unit can support up to 4 PMS.
You can also upload a configuration file to use a 3rd party PMS in the PMS UI (see below).
This makes the job simpler to online the sensors and to manage the sensors, instead of using virtual sensors.
This feature is licensed separately but our AKCP PMS are free to use.

VERY IMPORTANT NOTE: You need to connect the AC POWER TO THE PMS FIRST before connecting it to the sensorProbe+ unit and adding the meter to the web interface.

The PMS should ONLY be installed by a qualified Electrician Engineer who is also familiar with Modbus.

Please also see our AKCP 3 phase and single phase PMS manuals for more details and diagrams on our website that refer to the connections to our securityProbe base unit, however will also apply to the SP+ units.

“All Manuals” link then see section # 14:

- AC Power Monitoring Sensor 3 Phase
- AC Power Monitoring Sensor 3 Phase CT Quick Start Guide
- AC Power Monitoring Sensor Single Phase

RS485 Cabling and Run Length

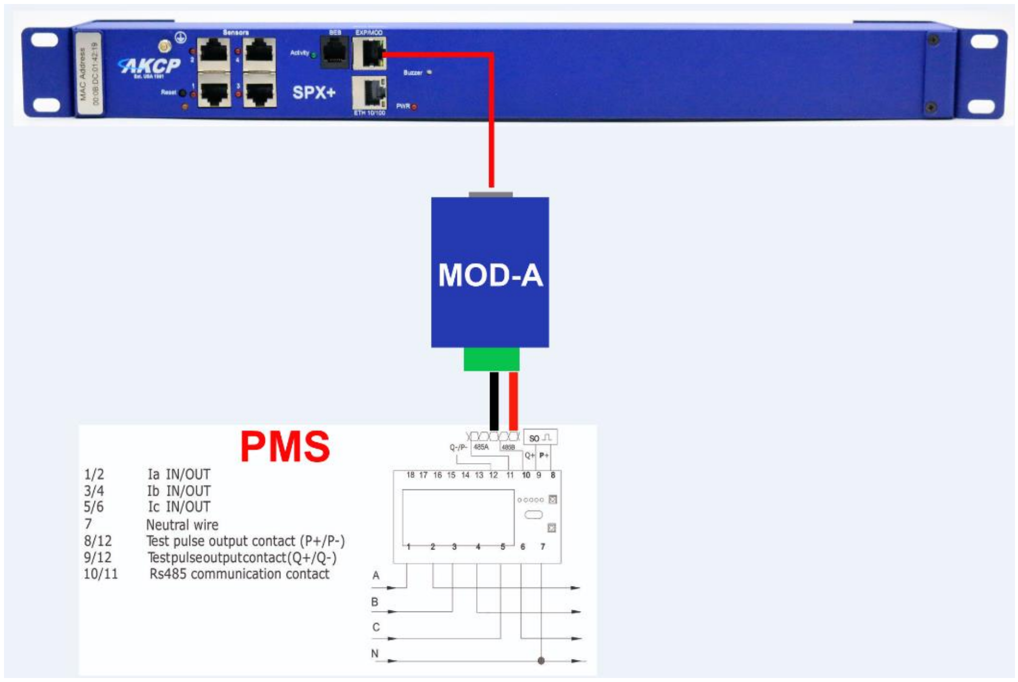
All Modbus manufacturers recommend a cable with 120Ohm characteristic impedance or higher so we highly recommend that you use shielded AWG 24 twisted pair copper wire and this cable has an impedance of at least 120Ohm, most importantly when connecting multiple PMS to a single base unit. The maximum cable length is determined by the combination of cable length (in meters) and data signaling rate. The data signaling rate on the power meter is 9600 bps (bits per second), so the maximum cable run length for this cable is roughly 300 meters, or 1000 feet.

It is possible to connect multiple PMS's to a single sensorProbe+ base unit however the more readings you require the slower the data update time will be. The maximum number of PMS / Modbus devices is 28 in the string without a repeater.

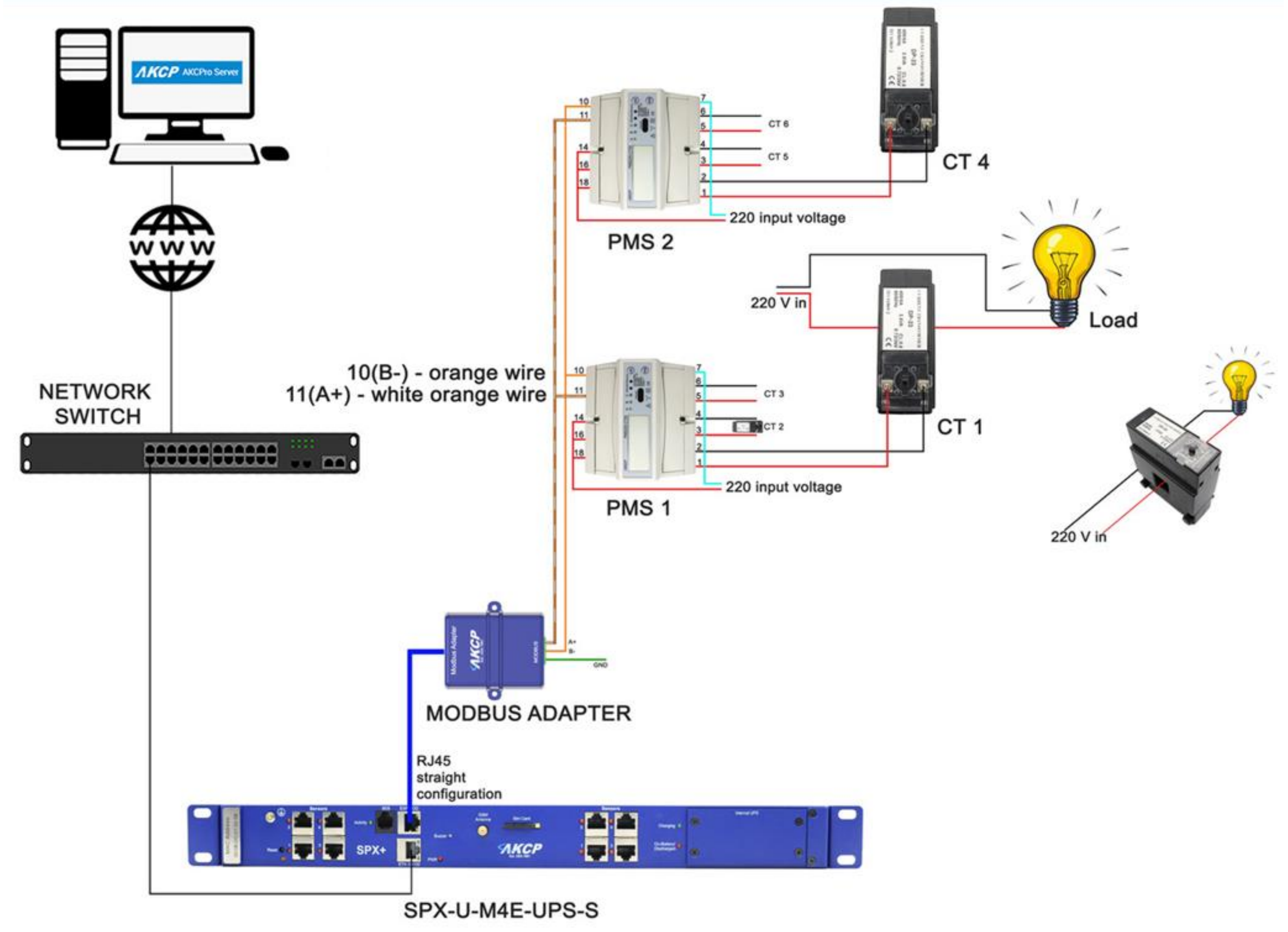
The AKCP PMS uses Modbus connection (RS485) through the Expansion port on the sensorProbe+ unit which is RJ45. You'll need to prepare a cable with the proper wiring (see wiring diagram in this manual) using a pair of wires from ports 23/24 on the PMS.

VERY IMPORTANT: On the interconnection between the TX+/- on the sensorProbe+ unit and the A B on the PMS, The A goes to - while the B goes to + .

AKCP Modbus Adapter: Using the MOD-A or the Modbus Adapter (see this manual for more details) to simplify the connection to the SP+. We highly recommend using this adapter when connecting any type of Modbus. See the image below for the wiring diagram and connection details.



1/2	Ia IN/OUT
3/4	Ib IN/OUT
5/6	Ic IN/OUT
7	Neutral wire
8/12	Test pulse output contact (P+/P-)
9/12	Testpulseoutputcontact(Q+/Q-)
10/11	Rs485 communication contact



Connecting CT’s (current transformers) to the AKCP PMS.

Up to 3 x CT’s can be connected to the AKCP CT Type PMS. The CT Type PMS & CT’s are designed and offered for requirments that include the monitoring of high current, or more than 100 Amps AC 3 phase.

Please check our website, datasheets and price list for the correct CT Type PMS and the CT's you require.

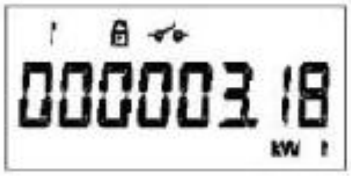
CT Ratio Settings

The CT ratio is dependent on the CT that was purchased, it has nothing to do with the amp rating of the circuit you are using.

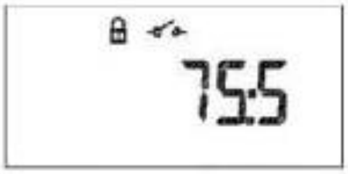
The CT’s that we offer are CTxxx/5A this means the ratio is 5A. If you purchase a CT with xxx/10A for example, then your ratio would be 10A.

How to set the current transformer ratio of the meter.

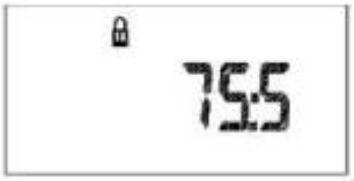
1. In about 12 seconds after powering up the energy meter, the padlock icon appears on screen. Press the PRG button (the switch icon should appear next to the padlock icon).



2. Within 10 seconds after pressing the PRG button, press the SEL button to cycle through the 27 different ratios. Transformers with similar rating should be physically connected with every energy meter.

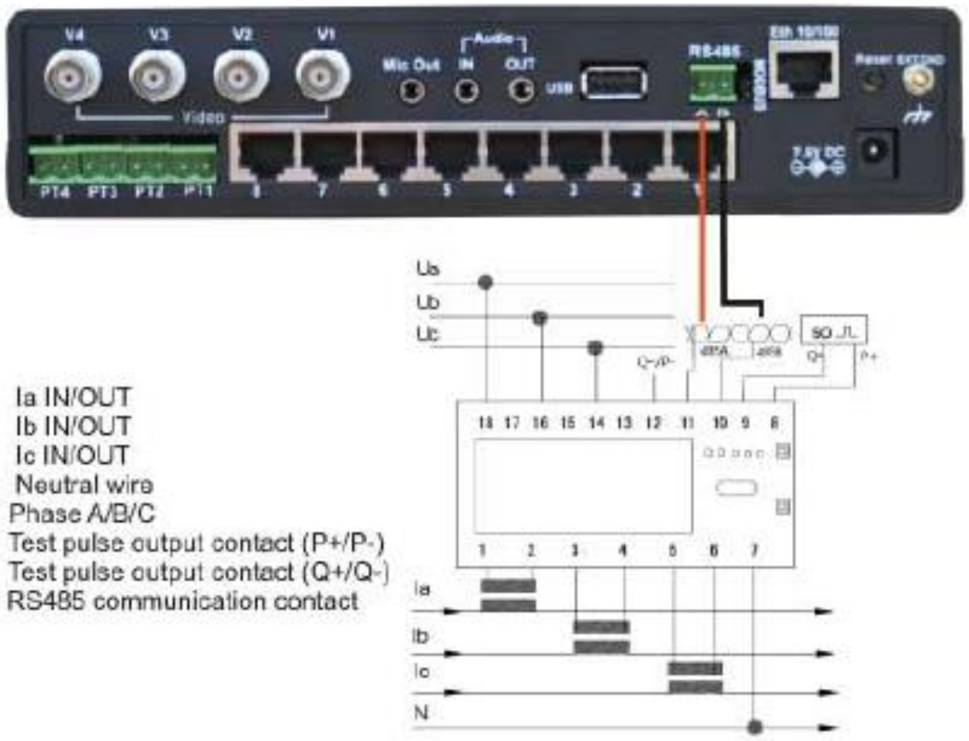


3. After choosing the correct ratio, press the PRG button to confirm. Setting the ratio is now complete and the switch button would disappear.



Note: If the lock icon is no longer visible on the screen, setting up the current transformer ratio is no longer allowed. You would have to power cycle the meter in able to program it.

Wiring diagram on how to connect the Energy Meter to securityProbe's RS485 port.



1/2	Ia IN/OUT
3/4	Ib IN/OUT
5/6	Ic IN/OUT
7	Neutral wire
18/16/14	Phase A/B/C
8/ 12	Test pulse output contact (P+/P-)
9/ 12	Test pulse output contact (Q+/Q-)
10/11	RS485 communication contact

Important Notes:

1. It is imperative that your securityProbe is updated to the latest version of firmware before installing the power meter. You can download this by logging into our web site here:
http://www.akcp.com/company/firmwareupdate.htm

2. This product has to be installed, maintained and repaired by a qualified electrician.

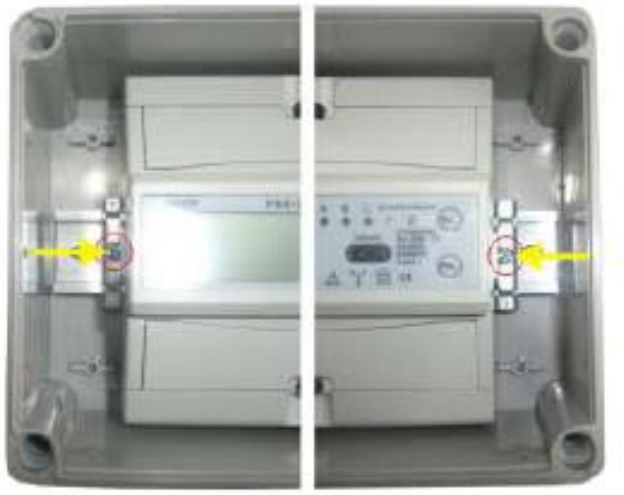
3. Prior to installation, carefully read the printed user manual.

4. The power meter has a Data transportation speed of 9600bps. RS422/485 electrical standards that do define the type of RS485 Cable/RS485 Connector recommends using 24 AWG twisted pair copper wire with an impedance of 120 and capacitance 16pF/ft for cables. Maximum length should not exceed 900 Meters.

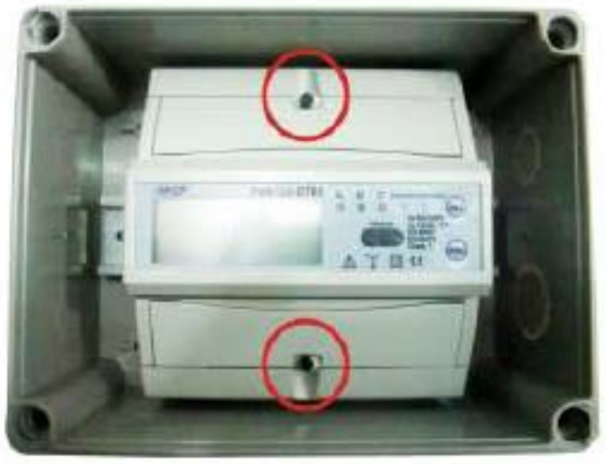
<u>IMPORTANT NOTE</u>: Pleaes refer to the Single Phase and 3 Phase AKCP PMS manuals on our website for all the details regarding the setup and connection of the PMS to our securityProbe base units as also noted above on page #24.

Steps on how to remove the Energy Meter from the DIN rail:

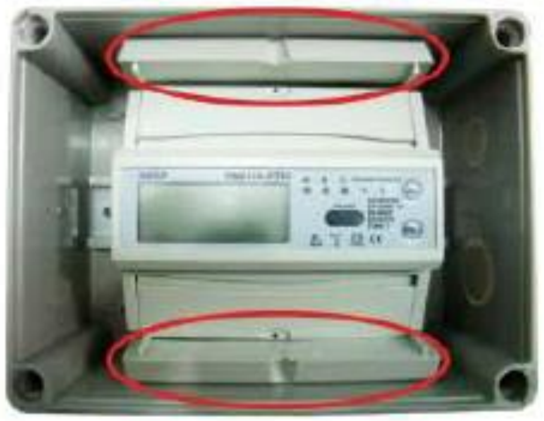
1. Loosen both screws (to which yellow arrows are pointing at).



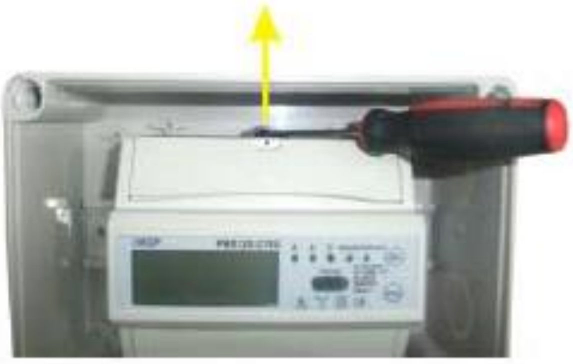
2. Pull the locking pins away from the energy meter.



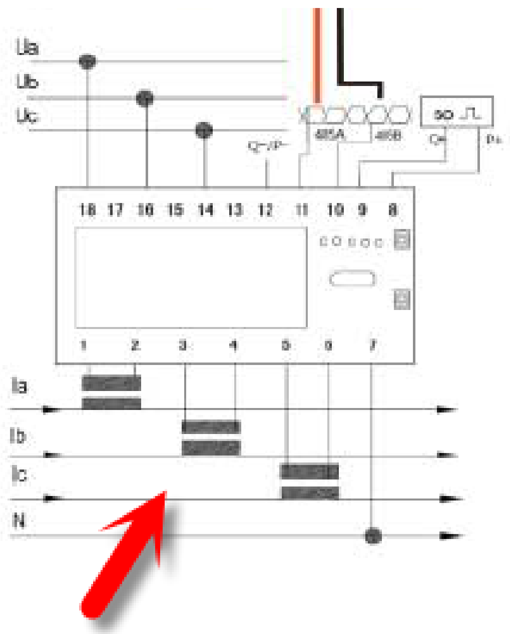
3. Slide out the top and bottom wiring shield using both hands.



4. Using a small screw driver, push the DIN rail locking mechanism upwards while pulling the energy meter out of the box.



1/2	Ia IN/OUT
3/4	Ib IN/OUT
5/6	Ic IN/OUT
7	Neutral wire
18/16/14	Phase A/B/C
8/ 12	Test pulse output contact (P+/P-)
9/ 12	Test pulse output contact (Q+/Q-)
10/11	RS485 communication contact



The 3 x CT’s connect to the 3 phase AKCP CT Type power meter as shwon in the diagram above. There are two wires on each CT that connect to the PMS inputs 1 & 2 CT #1, 3 & 4 CT #2 and 5 & 6 for CT #3. #7 is the neutral.

The CT clamps around the live wire, and detects the current in the cable by its magnetic field. Higher currents gives off a stronger magnetic field. The output from the CT goes to the power meter. The power meter will also be connected to a voltage source, so now it can calculate total power which is voltage x current.

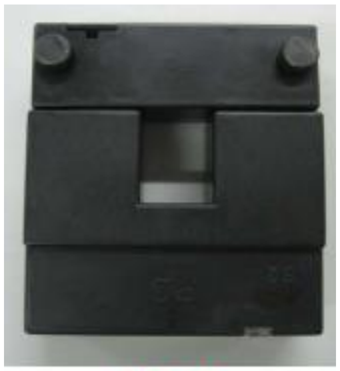
The AKCP 3 phase CT Type power meter works with the SP2+E, SPX+ and our securityProbe, so therefore the current transformers will also work with both base units.

Installation of the CT’s is recommended to be done by a qualified electrician. Although the installation is quite straight forward, it does require some knowledge of electronics. For example, tapping into a voltage source, and ensuring the CT is clamped around the live wire. You can not simply clamp the CT around the complete cable as in AC current you have two wires live and neutral that will cancel each other out as the current alternates. So you must ensure that you have access to the live wire only. Again, we can’t stress enough that the installation of the AKCP, or other 3rd party power meters be installed ONLY by a qualified professional electrician.

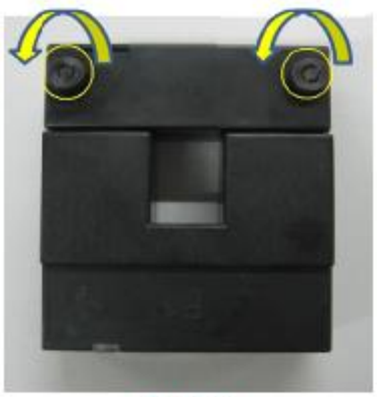
Handling Steps for the AKCP CT’s (Current Transformers)

It is very important to follow the steps carefully to prevent damage to the CT’s which would void the warranty.

1 . There are 2 bolts wich are holding the upper and lower pieces of the transformer together. The first step is to locate the circular nuts, then turn them counterclockwise until fully detached. As shown in the picture below (bolt side of the CT).

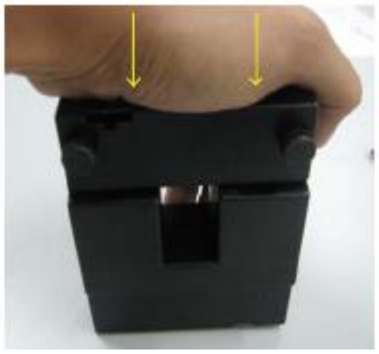


BOLT SIDE

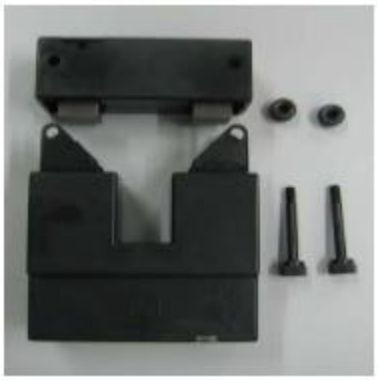


NUT SIDE

2. Position the transformer on a non slip surface then depress it firmly. Pull out each bolt carefully before releasing it. See image below.

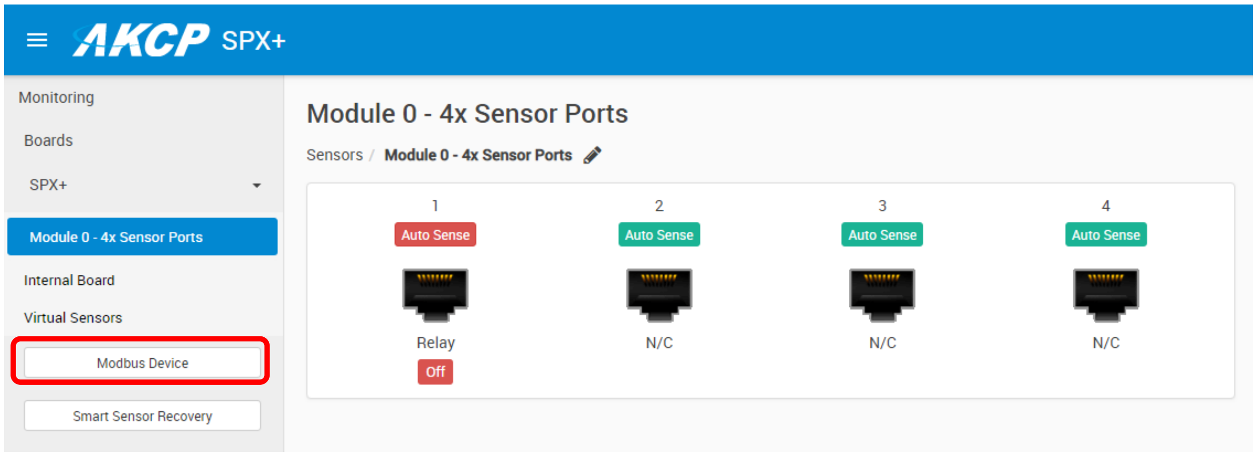


3. To reassemble the transformer, also remember to depress it firmly once again before reinserting the 2 bolts back in place. See image below.

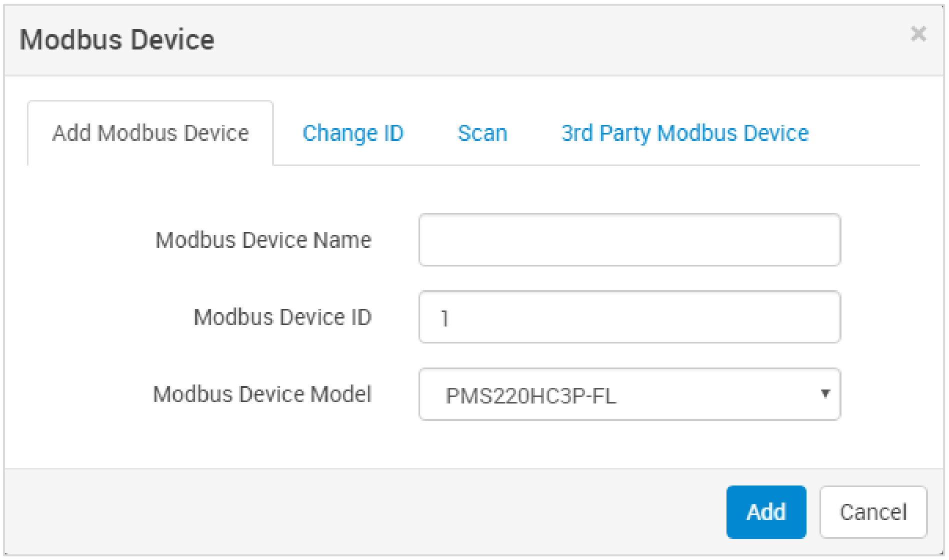


Adding the AKCP PMS to the sensorProbe+ web interface

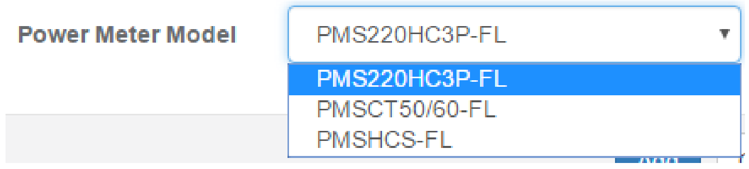
Once you have connected your PMS unit to the sensorProbe+ and you have ensured correct power connection to the PMS, go to the Sensors menu and click on the **Modbus Device** button.



This will bring up a popup window where you can input the meter's name, ID and model.

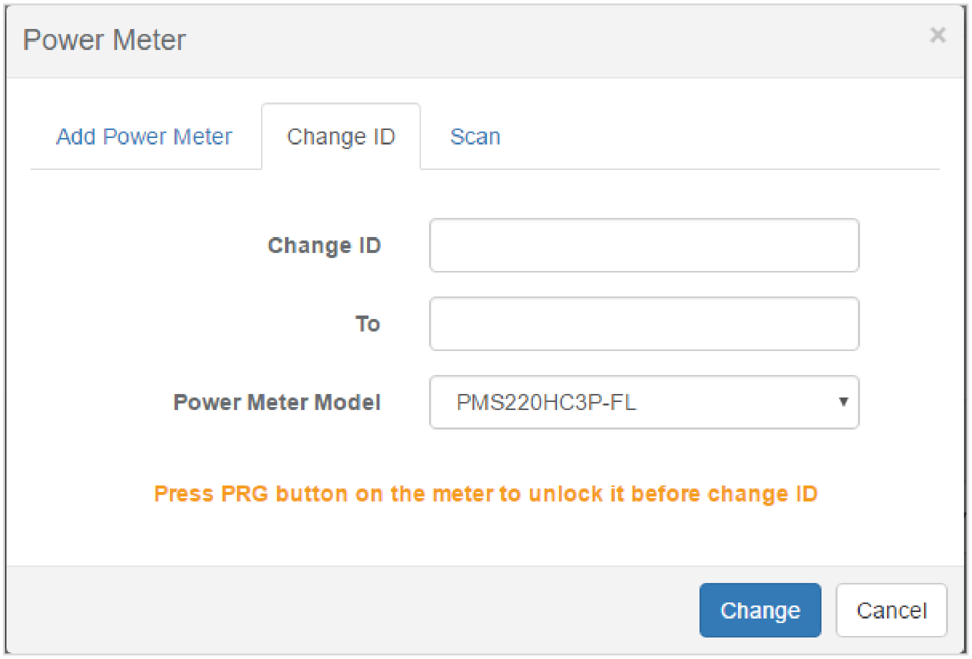


Modbus Device Name:
Modbus Device ID: 1
Modbus Device Model: PMS220HC3P-FL



Currently 3 types of AKCP power meters are supported:
PMSHCS-FL - Single Phase Meter
PMS220HC3P-FL - 3 Phase Meter
PMSCT50/60-FL - 3 Phase Meter CT Type

In case you need to change the ID of a PMS, you can do so on the Change ID tab.


Power Meter

Add Power Meter | Change ID | Scan

Change ID

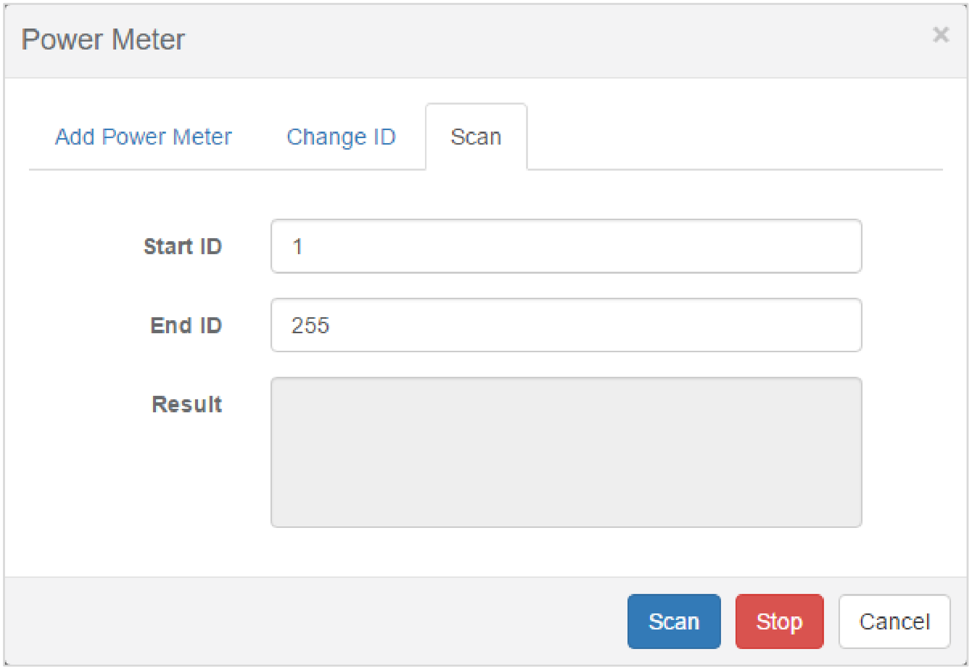
To

Power Meter Model: PMS220HC3P-FL

Press PRG button on the meter to unlock it before change ID

Change | Cancel

If you don't know the ID of a PMS that you've connected, you can run a scan for it on the Scan tab.


Power Meter

Add Power Meter | Change ID | Scan

Start ID: 1

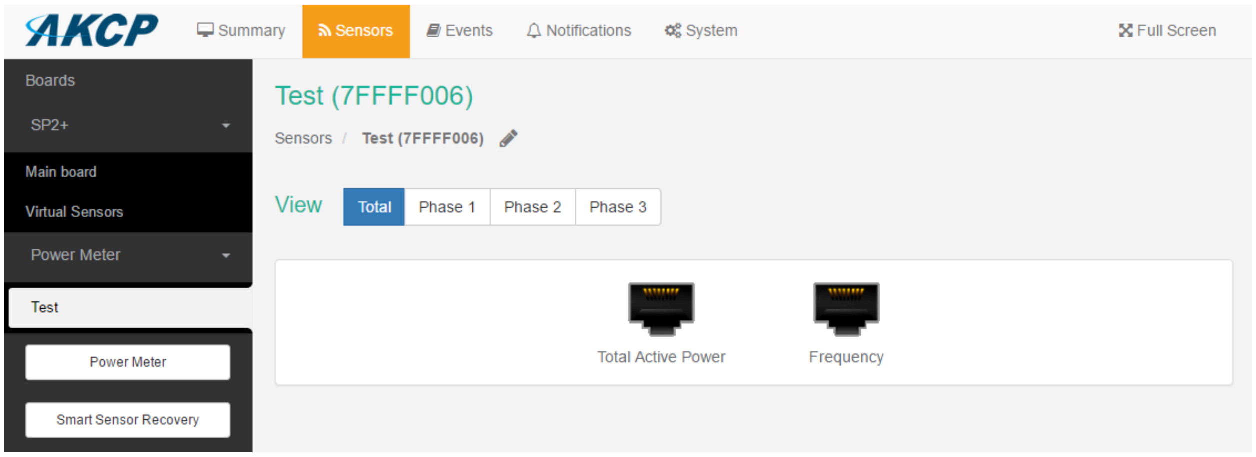
End ID: 255

Result

Scan | Stop | Cancel

Once the scan has begun any connected PMS units will be detected and added. You can enter your Scan range if you know the amount of meters you will be adding.You can stop the scan at any time by clicking on the "Stop" button.

The PMS will become online automatically and you can begin to configure it.

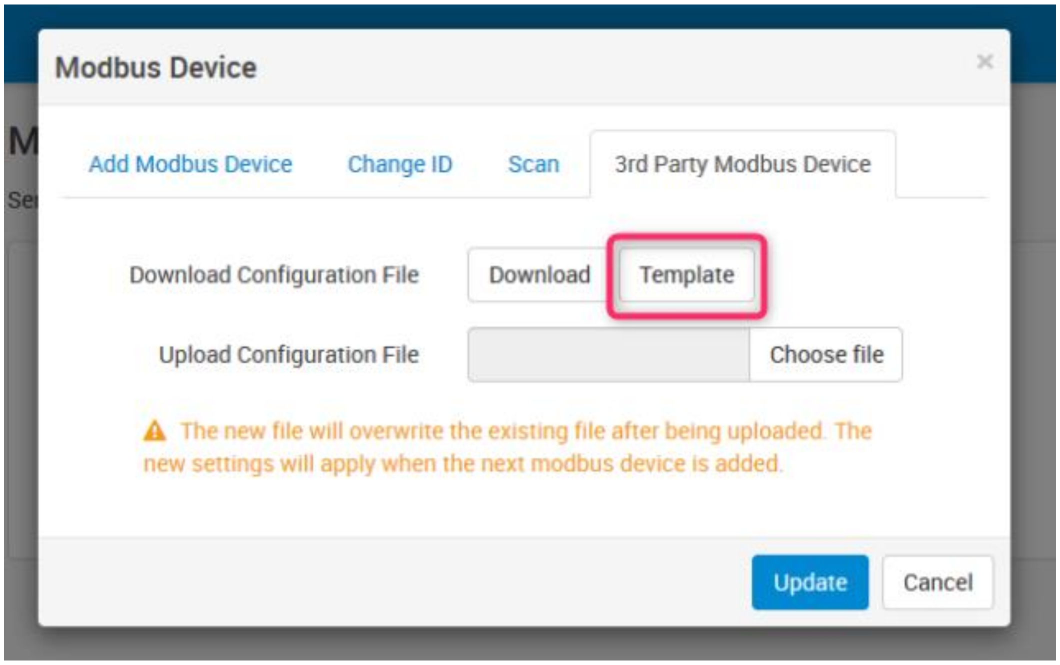


When adding multiple PMS to the web interface you have to add each PMS one at a time then change the ID from "1" to a different ID. Scan again then change "1" to a unique ID (like PMS1, PMS2) and so on. Repeat this procedure with all PMS's you plan on connecting.
Note: This will guarantee all of the PMS's in the daisy chain will function normally without any problem and insure there are no duplicate PMS ID numbers in the chain.

After you have completed the pre-configurations of all of your PMS's that you are going to connect, then reconnect each of them again in the same numerical sequence as you previously configured them with. Start with PMS ID 1 and PMS ID 2. If these are working OK, then you can connect the other PMS's in the string.
Note: Connect PMS ID 1 and PMS ID 2 on the same bus.

Adding 3rd Party PMS

On newer SP+ firmware (since 2018) you'll also get an option to use any other 3rd Party Modbus Device.
Note: This feature requires a separate license.


Modbus Device

Download Configuration File

Upload Configuration File

The new file will overwrite the existing file after being uploaded. The new settings will apply when the next modbus device is added.

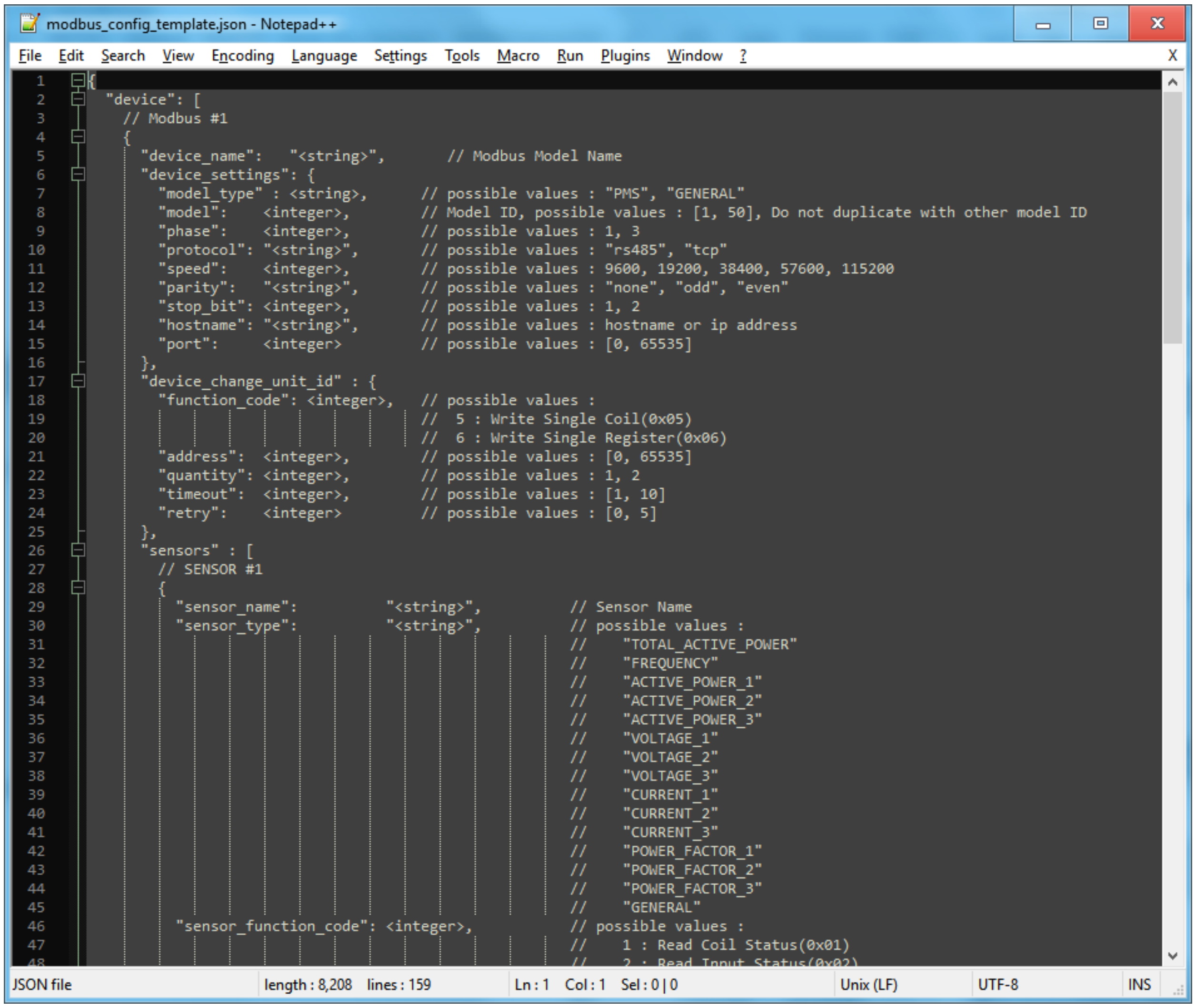
You'll have to create your own Json file for your PMS from the provided template file.

Click on the **Template** button. This will download a Json file, which you need to edit to configure the sensors you want to poll, and then upload it back to the unit.
The template includes information fields for 2 devices with 2 sensors each.

To edit this file, you'll need to use Notepad++ or similar editor that supports the Linux file format.
If the file has errors or unclosed sections, the unit cannot parse the configuration.
Therefore edit the file carefully and make sure each section opening has a closing.

We'll provide 2 example configurations below, that you can use to compare with the template file and make your own.
Upload it with the **Choose file** button, browse to your file and then click **Update**.

After you've uploaded your custom configuration, you can re-download it again with the **Download** button.


modbus_config_template.json - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
JSON file
length : 8,208 lines : 159
Ln : 1 Col : 1 Sel : 0 | 0
Unix (LF)
UTF-8
INS

```
{
  "device": [
    // Modbus #1
    {
      "device_name":   "<string>",       // Modbus Model Name
      "device_settings": {
        "model_type" : <string>,      // possible values : "PMS", "GENERAL"
        "model":    <integer>,        // Model ID, possible values : [1, 50], Do not duplicate with other model ID
        "phase":    <integer>,        // possible values : 1, 3
        "protocol": "<string>",       // possible values : "rs485", "tcp"
        "speed":    <integer>,        // possible values : 9600, 19200, 38400, 57600, 115200
        "parity":   "<string>",       // possible values : "none", "odd", "even"
        "stop_bit": <integer>,        // possible values : 1, 2
        "hostname": "<string>",       // possible values : hostname or ip address
        "port":     <integer>         // possible values : [0, 65535]
      },
      "device_change_unit_id" : {
        "function_code": <integer>,   // possible values :
                                      //  5 : Write Single Coil(0x05)
                                      //  6 : Write Single Register(0x06)
        "address":  <integer>,        // possible values : [0, 65535]
        "quantity": <integer>,        // possible values : 1, 2
        "timeout":  <integer>,        // possible values : [1, 10]
        "retry":    <integer>         // possible values : [0, 5]
      },
      "sensors" : [
        // SENSOR #1
        {
          "sensor_name":          "<string>",         // Sensor Name
          "sensor_type":          "<string>",         // possible values :
                                                      //    "TOTAL_ACTIVE_POWER"
                                                      //    "FREQUENCY"
                                                      //    "ACTIVE_POWER_1"
                                                      //    "ACTIVE_POWER_2"
                                                      //    "ACTIVE_POWER_3"
                                                      //    "VOLTAGE_1"
                                                      //    "VOLTAGE_2"
                                                      //    "VOLTAGE_3"
                                                      //    "CURRENT_1"
                                                      //    "CURRENT_2"
                                                      //    "CURRENT_3"
                                                      //    "POWER_FACTOR_1"
                                                      //    "POWER_FACTOR_2"
                                                      //    "POWER_FACTOR_3"
                                                      //    "GENERAL"
          "sensor_function_code": <integer>,          // possible values :
                                                      //    1 : Read Coil Status(0x01)
                                                      //    2 : Read Input Status(0x02)
```

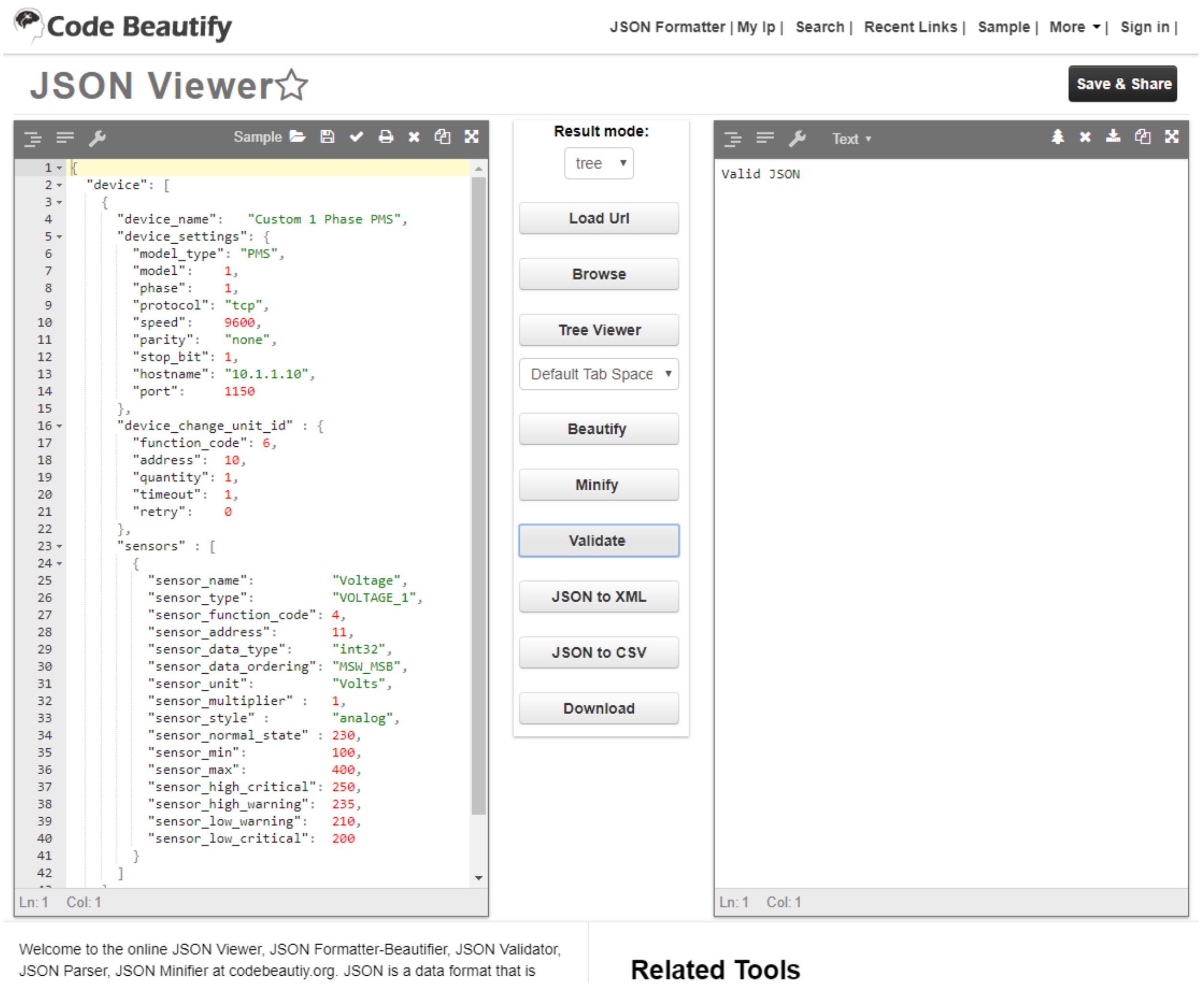
Edit the file as per your Modbus device requirements, and save the file with a different name.
This way you can always return to the template for correct examples.

You can comment out the unused values or lines with // characters (as seen for the help text).
We advise you to not remove lines, only comment them out.

You must put any string values in double quotes as "string" and any integer without quotes.

If you only want to add 1 PMS or 1 Modbus device with the template file, then you can remove the second entry from the end of the file.
This starts with // Modbus #2 and so on separated by comma (,).
Then you'll need to change the ending }, to be }
Notepad++ can help you to track the file sections.

We recommend you to check the edited Json file with an online validator, so you could be sure that your code has no errors:
https://codebeautify.org/jsonviewer



```
{
  "device": [
    {
      "device_name":   "Custom 1 Phase PMS",
      "device_settings": {
        "model_type": "PMS",
        "model":    1,
        "phase":    1,
        "protocol": "tcp",
        "speed":    9600,
        "parity":   "none",
        "stop_bit": 1,
        "hostname": "10.1.1.10",
        "port":     1150
      },
      "device_change_unit_id" : {
        "function_code": 6,
        "address":  10,
        "quantity": 1,
        "timeout":  1,
        "retry":    0
      },
      "sensors" : [
        {
          "sensor_name":          "Voltage",
          "sensor_type":          "VOLTAGE_1",
          "sensor_function_code": 4,
          "sensor_address":       11,
          "sensor_data_type":     "int32",
          "sensor_data_ordering": "MSW_MSB",
          "sensor_unit":          "Volts",
          "sensor_multiplier" :   1,
          "sensor_style" :        "analog",
          "sensor_normal_state" : 230,
          "sensor_min":           100,
          "sensor_max":           400,
          "sensor_high_critical": 250,
          "sensor_high_warning":  235,
          "sensor_low_warning":   210,
          "sensor_low_critical":  200
        }
      ]
```

```
Valid JSON
```

Welcome to the online JSON Viewer, JSON Formatter-Beautifier, JSON Validator, JSON Parser, JSON Minifier at codebeautiy.org. JSON is a data format that is

Load your configuration in the left pane and click on the **Validate** button.
If there are no errors, it will just say “Valid JSON” and you can upload it to the unit.
If there’s mistype or missing parts of the code, the validator will show the exact lines of code where the error is.

Note: AKCP does not support and is not responsible for the usage of this third party tool. It is only provided for user convenience.

Example 1

An example configuration is provided below for 1 PMS, with 1 Phase and 1 voltage sensor (the comments are removed for easier view):

```
{
  "device": [
    {
      "device_name":   "Custom 1 Phase PMS",
      "device_settings": {
        "model_type": "PMS",
        "model":    1,
        "phase":    1,
        "protocol": "tcp",
        "speed":    9600,
        "parity":   "none",
        "stop_bit": 1,
        "hostname": "10.1.1.10",
        "port":     1150
      },
      "device_change_unit_id" : {
        "function_code": 6,
        "address":  10,
        "quantity": 1,
        "timeout":  1,
        "retry":    0
      },
      "sensors" : [
        {
          "sensor_name":          "Voltage",
          "sensor_type":          "VOLTAGE_1",
          "sensor_function_code": 4,
          "sensor_address":       11,
          "sensor_data_type":     "int32",
          "sensor_data_ordering": "MSW_MSB",
          "sensor_unit":          "Volts",
          "sensor_multiplier" :   1,
          "sensor_style" :        "analog",
          "sensor_normal_state" : 230,
          "sensor_min":           100,
          "sensor_max":           400,
          "sensor_high_critical": 250,
          "sensor_high_warning":  235,
          "sensor_low_warning":   210,
          "sensor_low_critical":  200
        }
      ]
    }
  ]
}
```

Example 2

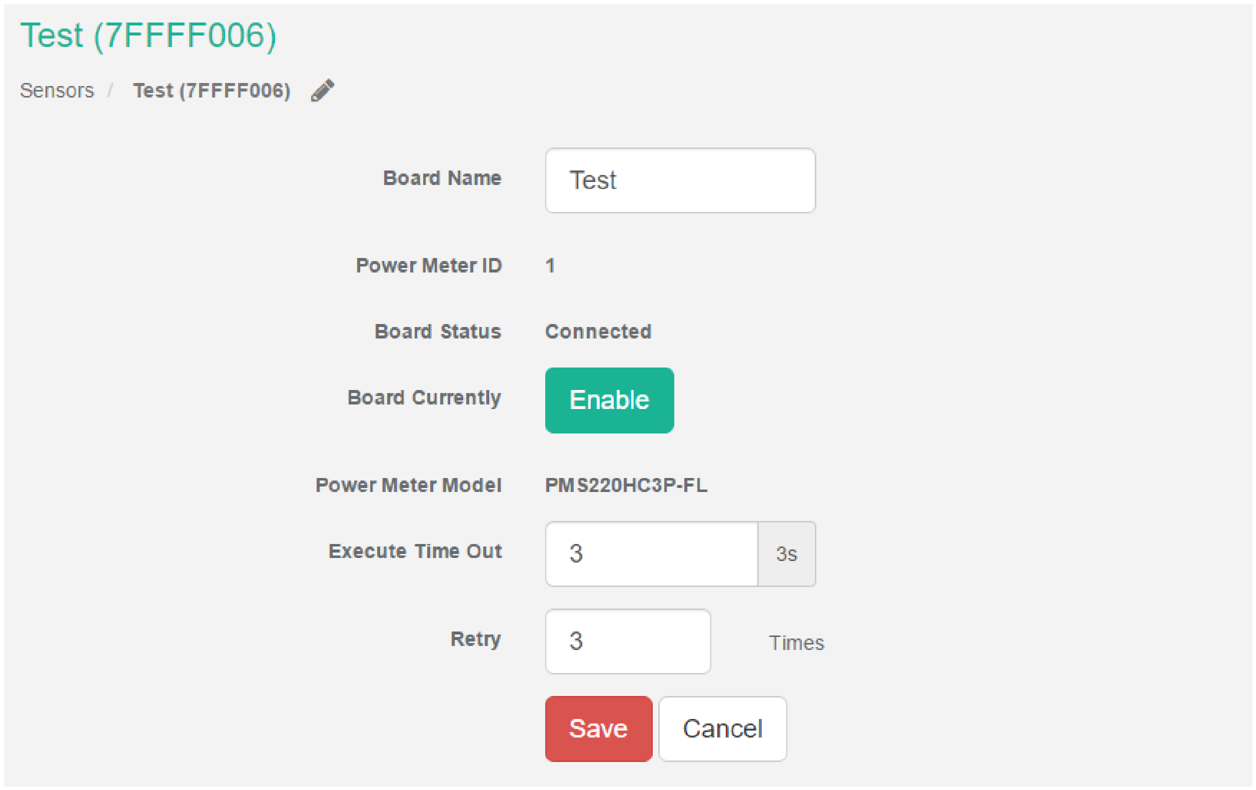
Below you can see the configuration of a preconfigured 3 Phase AKCP PMS model ALED3D5FD10:

```
{
  "device": [
  {
    "device_name" : "ALED3D5FD10",
    "device_settings" :
    {
      "model" : 3,
      "phase" : 3,
      "protocol" : "rs485",
      "speed" : 9600,
      "parity": "even",
      "stop_bit": 1
    },
    "sensors" :
    [
      {
        "sensor_name" : "TOTAL_ACTIVE_POWER_PORT",
        "sensor_type" : "TOTAL_ACTIVE_POWER",
        "sensor_function_code" : "4",
        "sensor_address" : "512",
        "sensor_data_type" : "float",
        "sensor_data_ordering" : "MSW_MSB",
        "sensor_unit" : "kWh",
        "sensor_factor" : "10",
        "sensor_min" : -1,
        "sensor_max" : "10000000",
        "sensor_high_critical" : "10000000",
        "sensor_high_warning" : "10000000",
        "sensor_low_warning" : "-1",
        "sensor_low_critical" : "-1"
      },
      {
        "sensor_name" : "FREQUENCY_PORT",
        "sensor_type" : "FREQUENCY",
        "sensor_function_code" : "4",
        "sensor_address" : "1560",
        "sensor_data_type" : "float",
        "sensor_data_ordering" : "MSW_MSB",
        "sensor_unit" : "Hz",
        "sensor_factor" : "100",
        "sensor_min" : "0",
        "sensor_max" : "100",
        "sensor_high_critical" : "80",
        "sensor_high_warning" : "70",
        "sensor_low_warning" : "30",
        "sensor_low_critical" : "20"
      },
      {
        "sensor_name" : "PHASE_LINE_1_ACTIVE_POWER_PORT",
        "sensor_type" : "ACTIVE_POWER_1",
        "sensor_function_code" : "4",
        "sensor_address" : "1536",
        "sensor_data_type" : "float",
        "sensor_data_ordering" : "MSW_MSB",
        "sensor_unit" : "kW",
        "sensor_factor" : "1000",
        "sensor_min" : "0",
        "sensor_max" : "1000",
        "sensor_high_critical" : "800",
        "sensor_high_warning" : "600",
        "sensor_low_warning" : "400",
        "sensor_low_critical" : "200"
      },
      {
        "sensor_name" : "PHASE_LINE_1_VOLTAGE_PORT",
        "sensor_type" : "VOLTAGE_1",
        "sensor_function_code" : "4",
        "sensor_address" : "1542",
        "sensor_data_type" : "float",
        "sensor_data_ordering" : "MSW_MSB",
        "sensor_unit" : "V",
        "sensor_factor" : "10",
        "sensor_min" : "0",
        "sensor_max" : "400",
        "sensor_high_critical" : "320",
        "sensor_high_warning" : "240",
        "sensor_low_warning" : "160",
        "sensor_low_critical" : "80"
      },
      {
        "sensor_name" : "PHASE_LINE_1_CURRENT_PORT",
        "sensor_type" : "CURRENT_1",
        "sensor_function_code" : "4",
        "sensor_address" : "1548",
        "sensor_data_type" : "float",
        "sensor_data_ordering" : "MSW_MSB",
        "sensor_unit" : "A",
        "sensor_factor" : "100",
        "sensor_min" : "0",
        "sensor_max" : "100",
        "sensor_high_critical" : "80",
        "sensor_high_warning" : "60",
        "sensor_low_warning" : "40",
        "sensor_low_critical" : "20"
      },
      {
        "sensor_name" : "PHASE_LINE_1_POWER_FACTOR_PORT",
        "sensor_type" : "POWER_FACTOR_1",
        "sensor_function_code" : "4",
        "sensor_address" : "1554",
      "sensor_data_type" : "float",
      "sensor_data_ordering" : "MSW_MSB",
      "sensor_unit" : "",
      "sensor_factor" : "100",
      "sensor_min" : "-1",
      "sensor_max" : "1",
      "sensor_high_critical" : "0.6",
      "sensor_high_warning" : "0.2",
      "sensor_low_warning" : "-0.2",
      "sensor_low_critical" : "-0.6"
    },
    {
      "sensor_name" : "PHASE_LINE_2_ACTIVE_POWER_PORT",
      "sensor_type" : "ACTIVE_POWER_2",
      "sensor_function_code" : "4",
      "sensor_address" : "1538",
      "sensor_data_type" : "float",
      "sensor_data_ordering" : "MSW_MSB",
      "sensor_unit" : "kW",
      "sensor_factor" : "1000",
      "sensor_min" : "0",
      "sensor_max" : "1000",
      "sensor_high_critical" : "800",
      "sensor_high_warning" : "600",
      "sensor_low_warning" : "400",
      "sensor_low_critical" : "200"
    },
    {
      "sensor_name" : "PHASE_LINE_2_VOLTAGE_PORT",
      "sensor_type" : "VOLTAGE_2",
      "sensor_function_code" : "4",
      "sensor_address" : "1544",
      "sensor_data_type" : "float",
      "sensor_data_ordering" : "MSW_MSB",
      "sensor_unit" : "V",
      "sensor_factor" : "10",
      "sensor_min" : "0",
      "sensor_max" : "400",
      "sensor_high_critical" : "320",
      "sensor_high_warning" : "240",
      "sensor_low_warning" : "160",
      "sensor_low_critical" : "80"
    },
    {
      "sensor_name" : "PHASE_LINE_2_CURRENT_PORT",
      "sensor_type" : "CURRENT_2",
      "sensor_function_code" : "4",
      "sensor_address" : "1550",
      "sensor_data_type" : "float",
      "sensor_data_ordering" : "MSW_MSB",
      "sensor_unit" : "A",
      "sensor_factor" : "100",
      "sensor_min" : "0",
      "sensor_max" : "100",
      "sensor_high_critical" : "80",
      "sensor_high_warning" : "60",
      "sensor_low_warning" : "40",
      "sensor_low_critical" : "20"
    },
    {
      "sensor_name" : "PHASE_LINE_2_POWER_FACTOR_PORT",
      "sensor_type" : "POWER_FACTOR_2",
      "sensor_function_code" : "4",
      "sensor_address" : "1556",
      "sensor_data_type" : "float",
      "sensor_data_ordering" : "MSW_MSB",
      "sensor_unit" : "",
      "sensor_factor" : "100",
      "sensor_min" : "-1",
      "sensor_max" : "1",
      "sensor_high_critical" : "0.6",
      "sensor_high_warning" : "0.2",
      "sensor_low_warning" : "-0.2",
      "sensor_low_critical" : "-0.6"
    },
    {
      "sensor_name" : "PHASE_LINE_3_ACTIVE_POWER_PORT",
      "sensor_type" : "ACTIVE_POWER_3",
      "sensor_function_code" : "4",
      "sensor_address" : "1540",
      "sensor_data_type" : "float",
      "sensor_data_ordering" : "MSW_MSB",
      "sensor_unit" : "kW",
      "sensor_factor" : "1000",
      "sensor_min" : "0",
      "sensor_max" : "1000",
      "sensor_high_critical" : "800",
      "sensor_high_warning" : "600",
      "sensor_low_warning" : "400",
      "sensor_low_critical" : "200"
    },
    {
      "sensor_name" : "PHASE_LINE_3_VOLTAGE_PORT",
      "sensor_type" : "VOLTAGE_3",
      "sensor_function_code" : "4",
      "sensor_address" : "1546",
      "sensor_data_type" : "float",
      "sensor_data_ordering" : "MSW_MSB",
      "sensor_unit" : "V",
      "sensor_factor" : "10",
      "sensor_min" : "0",
      "sensor_max" : "400",
      "sensor_high_critical" : "320",
      "sensor_high_warning" : "240",
      "sensor_low_warning" : "160",
      "sensor_low_critical" : "80"
```

```
      },
      {
        "sensor_name" : "PHASE_LINE_3_CURRENT_PORT",
        "sensor_type" : "CURRENT_3",
        "sensor_function_code" : "4",
        "sensor_address" : "1552",
        "sensor_data_type" : "float",
        "sensor_data_ordering" : "MSW_MSB",
        "sensor_unit" : "A",
        "sensor_factor" : "100",
        "sensor_min" : "0",
        "sensor_max" : "100",
        "sensor_high_critical" : "80",
        "sensor_high_warning" : "60",
        "sensor_low_warning" : "40",
        "sensor_low_critical" : "20"
      },
      {
        "sensor_name" : "PHASE_LINE_3_POWER_FACTOR_PORT",
        "sensor_type" : "POWER_FACTOR_3",
        "sensor_function_code" : "4",
        "sensor_address" : "1558",
        "sensor_data_type" : "float",
        "sensor_data_ordering" : "MSW_MSB",
        "sensor_unit" : "",
        "sensor_factor" : "100",
        "sensor_min" : "-1",
        "sensor_max" : "1",
        "sensor_high_critical" : "0.6",
        "sensor_high_warning" : "0.2",
        "sensor_low_warning" : "-0.2",
        "sensor_low_critical" : "-0.6"
      }
    ]
  }
 ]
}
```

PMS configuration

Each PMS will be treated as an expansion board. Click on the pencil icon next to the name of the meter to configure general parameters, rename it or disable it.



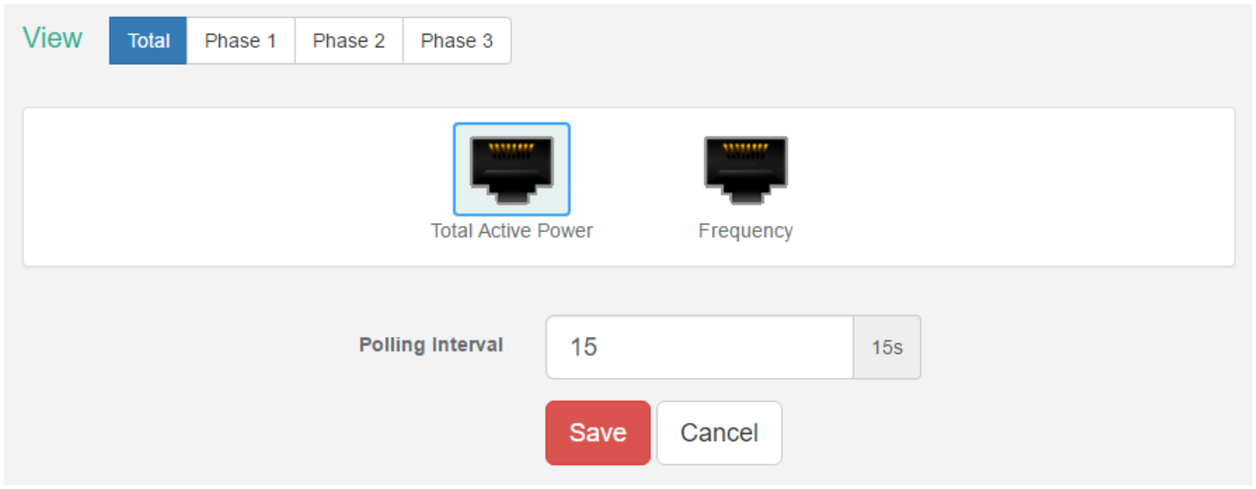
Test (7FFFF006)

Sensors / Test (7FFFF006)

Board Name	Test
Power Meter ID	1
Board Status	Connected
Board Currently	Enable
Power Meter Model	PMS220HC3P-FL
Execute Time Out	3 3s
Retry	3 Times

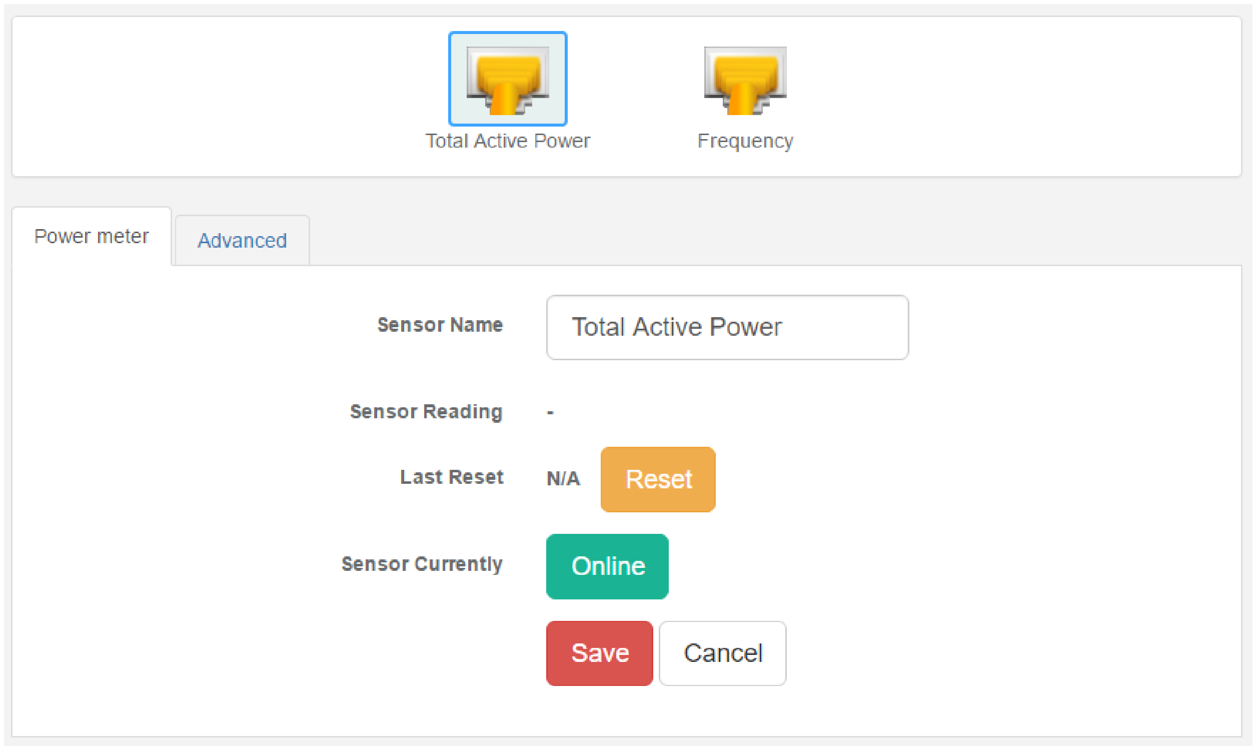
Note: you can only set the Execute Timeout and Retry parameters for all phases, not per phase as it's on the securityProbe series.

Before you'll see any reading from the meter, you'll need to configure its polling intervals.



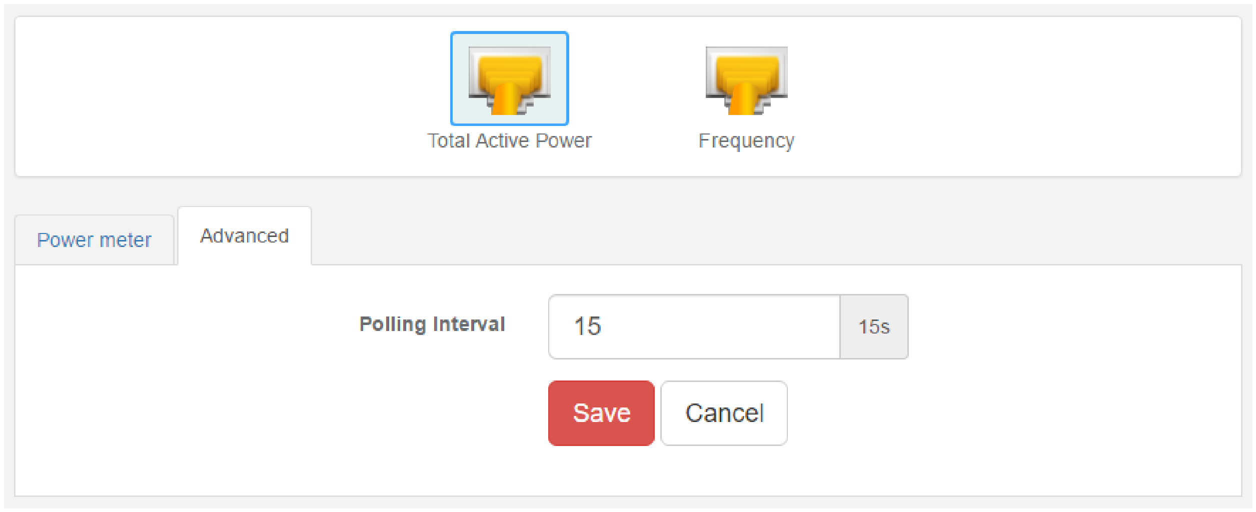
Polling Interval: 15 (15s)

You'll need to click on each sensor readings and save the polling interval, then you'll be able to: rename, view the sensor reading and take it offline.
In this example we show the Total Active Power.



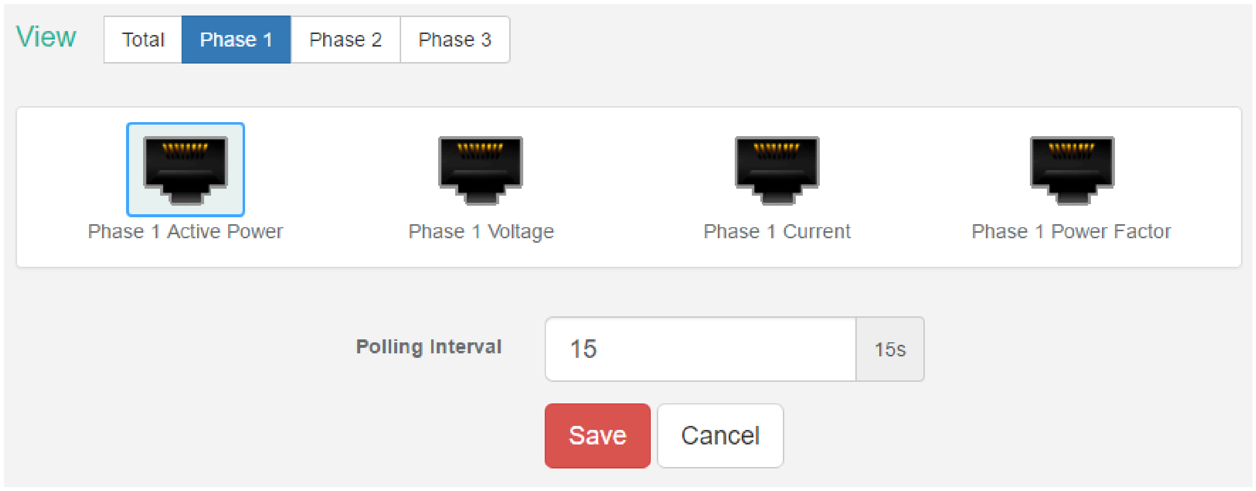
Sensor Name: Total Active Power
Sensor Reading: -
Last Reset: N/A
Sensor Currently: Online

You can modify the polling interval on the Advanced tab after you've put the sensor online.

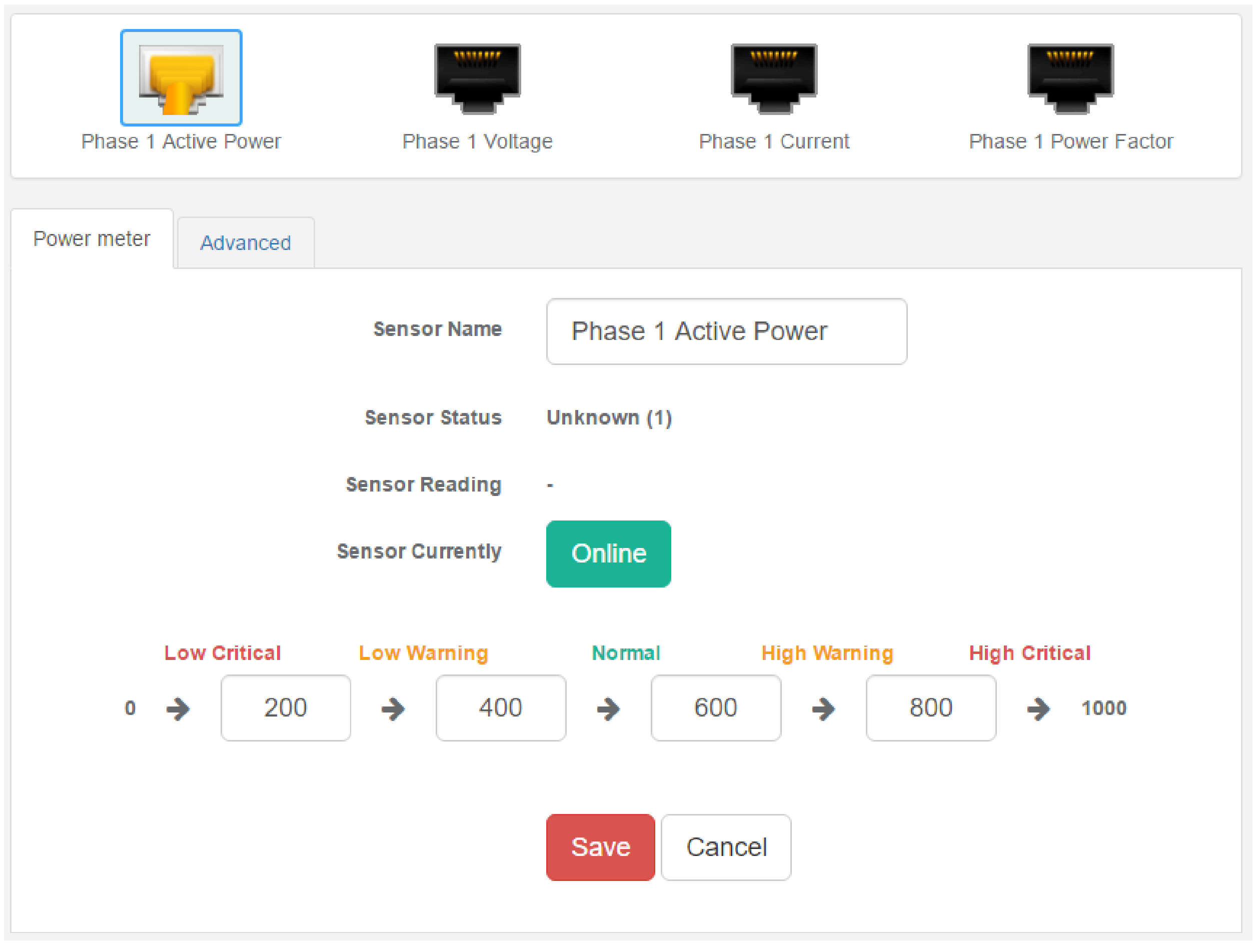


Polling Interval 15 15s

You'll need to manually put each sensor online for each phase you'll want to monitor, similarly to the Total Active Power reading.



Polling Interval 15 15s



Sensor Name: Phase 1 Active Power

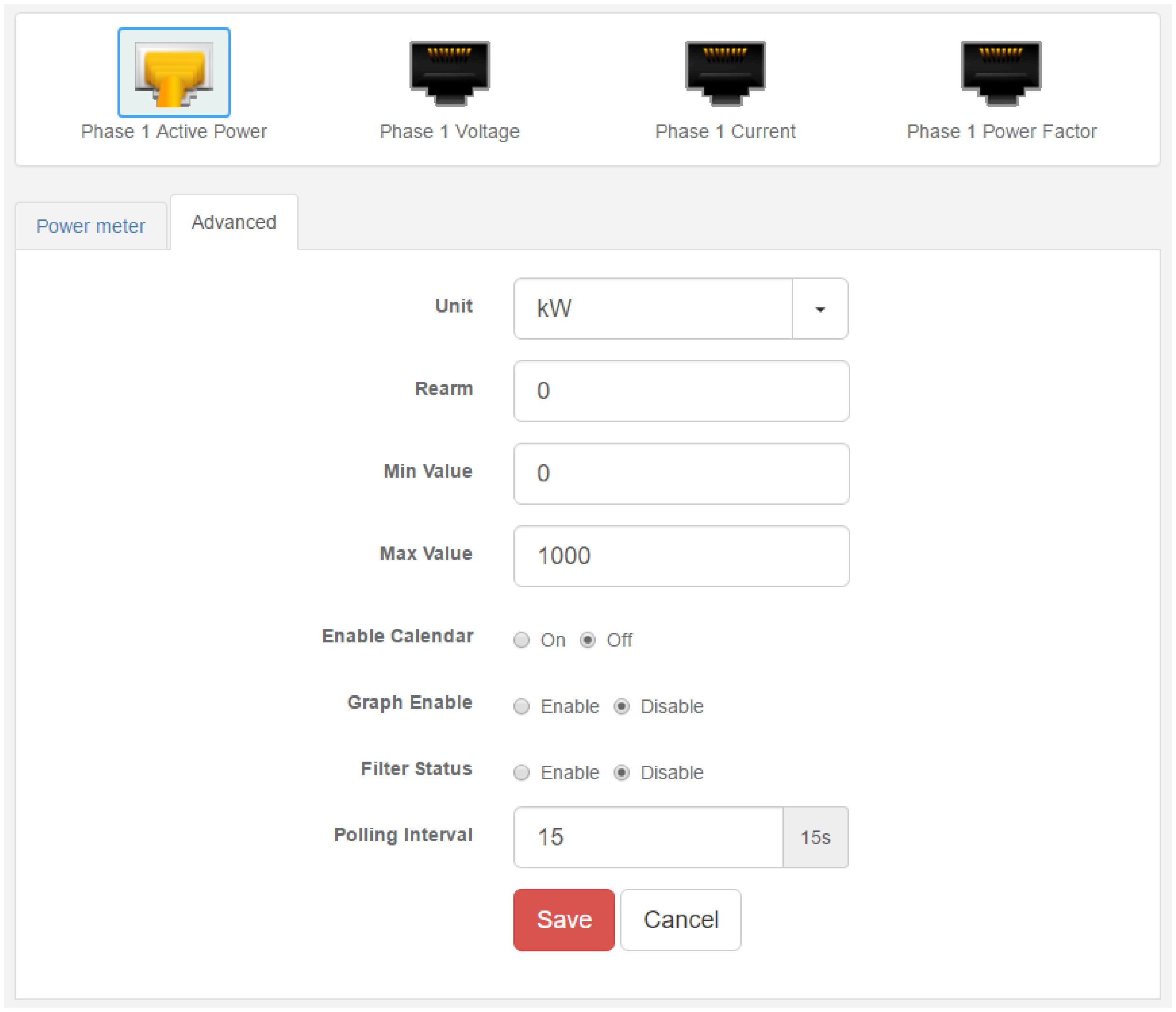
Sensor Status: Unknown (1)

Sensor Reading: -

Sensor Currently: Online

Low Critical	Low Warning	Normal	High Warning	High Critical
0 → 200	→ 400	→ 600	→ 800	→ 1000

For the Active Power reading, you can further define the warning and critical statuses.



Unit	kW
Rearm	0
Min Value	0
Max Value	1000
Enable Calendar	On [✓ Off]
Graph Enable	Enable [✓ Disable]
Filter Status	Enable [✓ Disable]
Polling Interval	15 15s

Under the Advanced tab, you can define similar parameters as with other AKCP sensors:

- Display unit
- Rearm
- Min/Max value
- Calendar and Graph support
- Filter status
- Polling interval

AKCP Modbus return values

Holding Register

The following is a list of what each of the return values from the Holding Register represent:

1 No Status (Sensor has not been initialized)
2 Normal
3 High Warning
4 High Critical
5 Low Warning
6 Log Critical
7 Sensor Error
8 Switch Low (Output)
9 Switch High (Output)

INPUT Register

If the sensor is OFFLINE, the INPUT register will return 32767
If the sensor is No Status or Sensor Error, the INPUT register is -32768

Name	**Type**
Temperature sensor	0x01
4-20ma sensor	0x02
Humidity sensor	0x03
Water detector sensor	0x04
Analog sensor	0x05
Security sensor	0x06
Airflow sensor	0x08
Siren sensor	0x09
Drycontact sensor	0x0A
AC voltage sensor	0x0C
Relay sensor	0x0D
Motion sensor	0x0E

Coil, discrete input, input register, holding register numbers and addresses

Some conventions govern how access to Modbus entities (coils, discrete inputs, input registers, holding registers) are referenced.

It is important to make a distinction between entity *number* and entity *address*:

- Entity *numbers* combine entity type and entity location within their description table
- Entity *address* is the starting address, a 16-bit value in the data part of the Modbus frame. As such its range goes from 0 to 65,535

In the traditional standard, *numbers* for those entities start with a digit, followed by a number of four digits in range 1–9,999:

- coils *numbers* start with a **zero** and then span from **0**0001 to **0**9999
- discrete input *numbers* start with a **one** and then span from **1**0001 to **1**9999
- input register *numbers* start with a **three** and then span from **3**0001 to **3**9999
- holding register *numbers* start with a **four** and then span from **4**0001 to **4**9999

This translates into *addresses* between 0 and 9,998 in data frames.
For example, in order to read holding registers starting at *number* 40001, corresponding *address* in the data frame will be 0 with a function code of 3 (as seen above). For holding registers starting at *number* 40100, *address* will be 99. Etc.

This limits the number of *addresses* to 9,999 for each entity. A *de facto* referencing extends this to the maximum of 65,536.
It simply consists of adding one digit to the previous list:

- coil *numbers* span from **0**00001 to **0**65536
- discrete input *numbers* span from **1**00001 to **1**65536
- input register *numbers* span from **3**00001 to **3**65536
- holding register *numbers* span from **4**00001 to **4**65536

When using the extended referencing, all *number* references must be exactly six digits. This avoids confusion between coils and other entities. For example, to know the difference between holding register #40001 and coil #40001, if coil #40001 is the target, it must appear as #040001.

Reading/writing ranges and explanations for each Modbus data type

16 bits unsigned int = 0...65535 (1 word)
16 bits signed int = -32768...+32767 (1 word)
32 bits unsigned int = 0...4 294 967 295 (2 words)
32 bits signed int = -2 147 483 648...+2 147 483 647 (2 words)
32 bits IEEE floating point = -3.4028E+38... +3.4028E+38 (2 words, Float32 type data: Single precision float with sign bit, 8 bits exponent, 23 bits mantissa (positive and negative normalized real))

Integer = any kind of numbers; the range depends on the selected data type as above

Boolean/Bool = 0 or 1 value (yes/no, true/false etc.)

Real = 32bit float value (as above), more info: https://store.chipkin.com/articles/how-real-floating-point-and-32-bit-data-is-encoded-in-modbus-rtu-messages

Word = 1 register value, 16bits of data, address between 0000 and 270E (as above)

U32 = 32-bit unsigned integer (2 words), number value between 0...4 294 967 295 (as above)

String = text value; Each register (16 bits) is interpreted as two ASCII characters (each 1 byte = 8 bits). Often 16 consecutive registers are used, allowing 32 characters in the string. Unicode/UTF-8 is typically not supported.

More information:
https://www.productinfo.schneider-electric.com/powertaglinkuserguide/powertag-link-user-guide/English/BM_PowerTag%20Link%20D%20User%20Manual_4af62430_T000501355.xml/$/TPC_ModbusTableFormatandDataTypes_4af62430_T000501590
https://minimalmodbus.readthedocs.io/en/stable/modbusdetails.html
https://www.simplymodbus.ca/FAQ.htm
https://help.campbellsci.com/CR6/Content/shared/Communication/Modbus/modbus-infostorage.htm?TocPath=Communications%20protocols%7CModbus%20communications%7C_____7

Please contact support@akcp.com if you have any further technical questions or problems.

Thanks for Choosing AKCP!