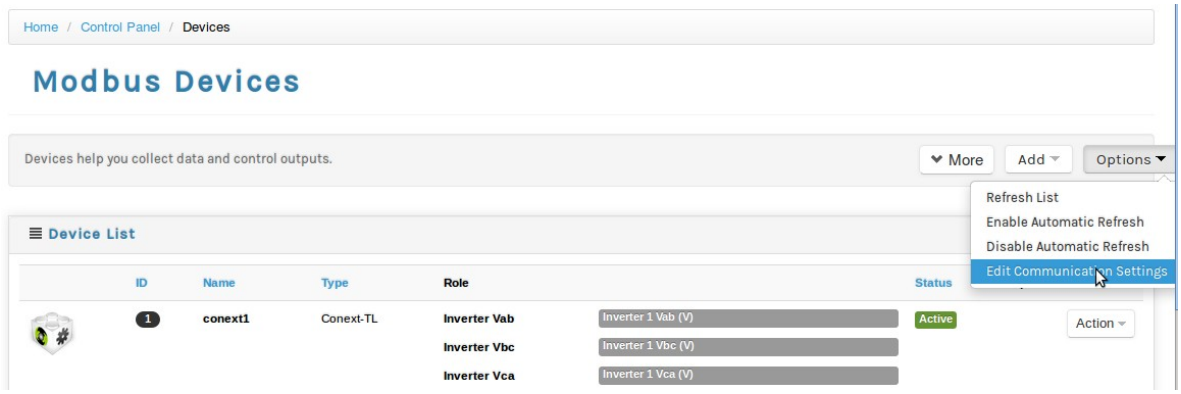
Rev 1.0 Jan 2015

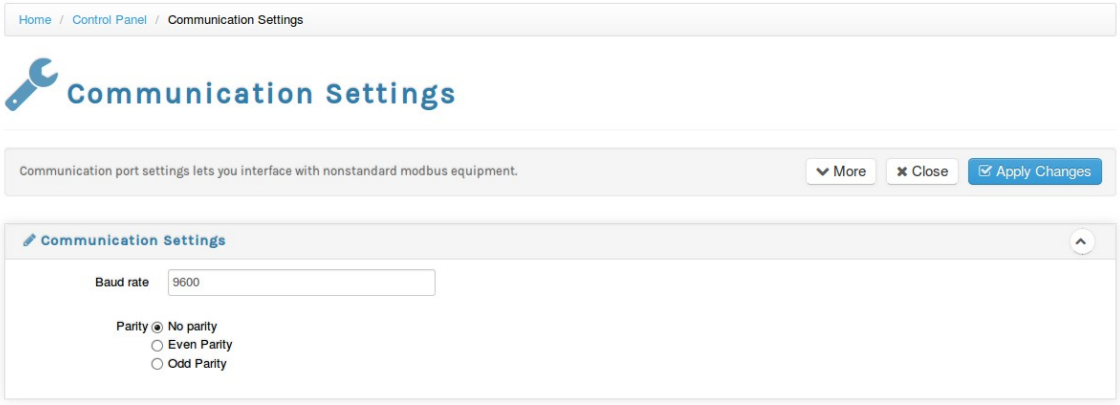
Configuring a Modbus RTU Driver to work with Wattmon

Wattmon can interface with any 3rd party device over Modbus RTU on RS-485. The baud rate of the device needs to be set to the same as the Wattmon, the default being 9600 bps, 8 data bits, no parity and 1 stop bit. Wattmon can be configured at a different baud rate and parity but it will mean that Wattmon specific modules such as the C752 will not function so use this only if you are working exclusively with 3rd party devices. Setting the baud rate can be done from the devices page.



Device page

In the communications page, you can change baud rate and parameters:



Understanding Modbus RTU

Modbus RTU is a protocol that allows multiple slave devices to interface with a master device (Wattmon) over a 2-wire bus. Modbus devices respond to four different types of data:

Digital inputs/Discrete Input Contacts

These are single bits and typically used on devices that have switches or other types of contacts whose status needs to be read

Digital Outputs/Discrete Output Coils

These are typically used to control relays or other contact devices

Read Only Registers/Analog Input Registers

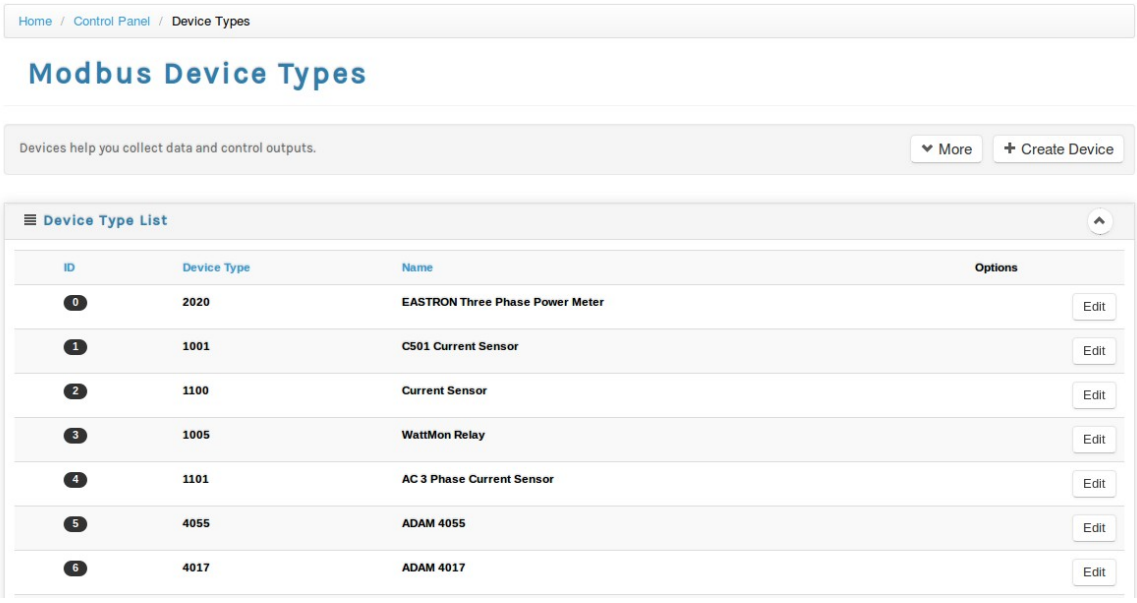
These are used to read system status from the modbus device. Each register has a specific address and value – refer to the documentation about the modbus device for a register map. Registers can be arranged in contiguous sequences or in groups of addresses – Wattmon drivers can handle single blocks or multiblock mode. Registers are 16 bits wide. However, often certain registers can be combined together to form 32-bit floating point or unsigned int values.

Read/Write Registers/Holding Registers

These are similar to read only registers, also 16-bits wide and with a unique address. However, these are accessed using a different function and allow values to be changed, making it possible to configure an external device over modbus.

Viewing Existing Devices

In the Control Panel, click Device Types to bring up the page as shown below:



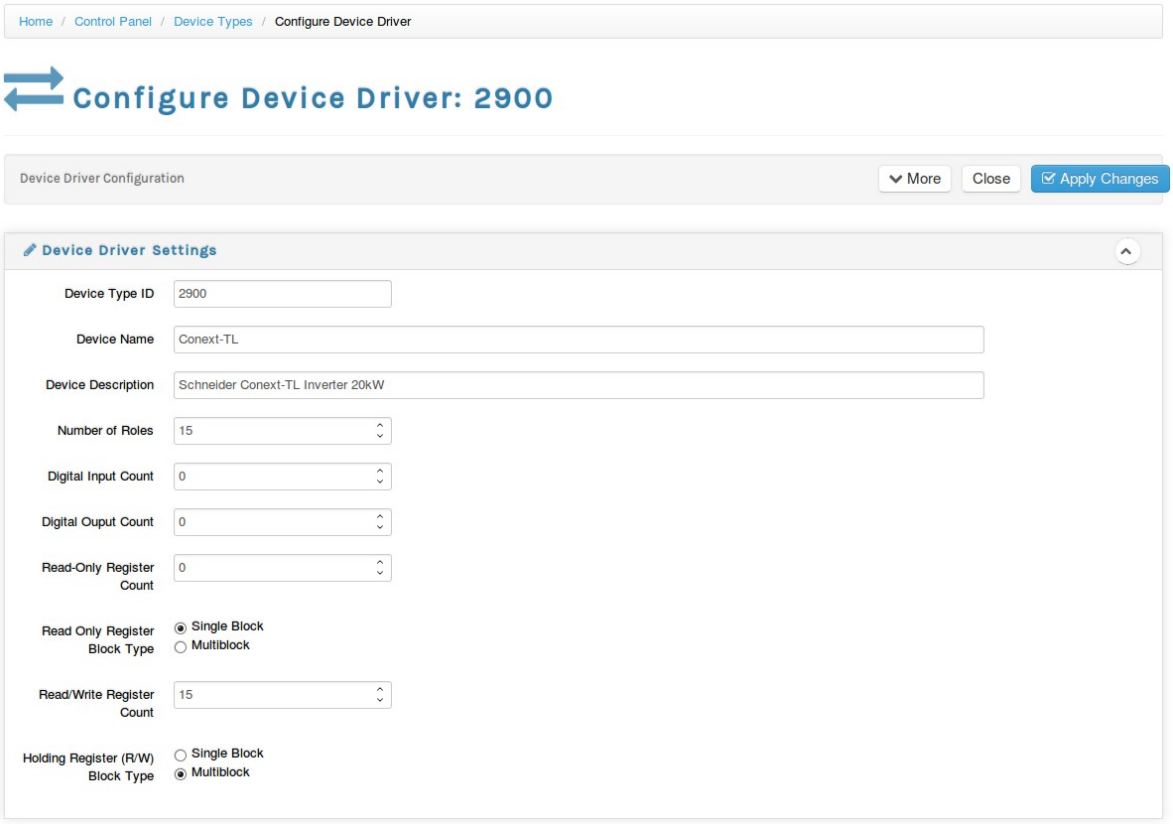
Device Types window

Each device in wattmon requires a unique device number – this is arbitrarily defined and should be in the range between 1000 and 3999. In order to edit an existing driver, click the Edit button in the list, or click Create Device to create a new device.

Configuring A Device Driver

The Configure device driver window has multiple sections. The steps will be defined in the following sections.

The *Device Driver Settings* page is shown below.



Device Driver Settings Page

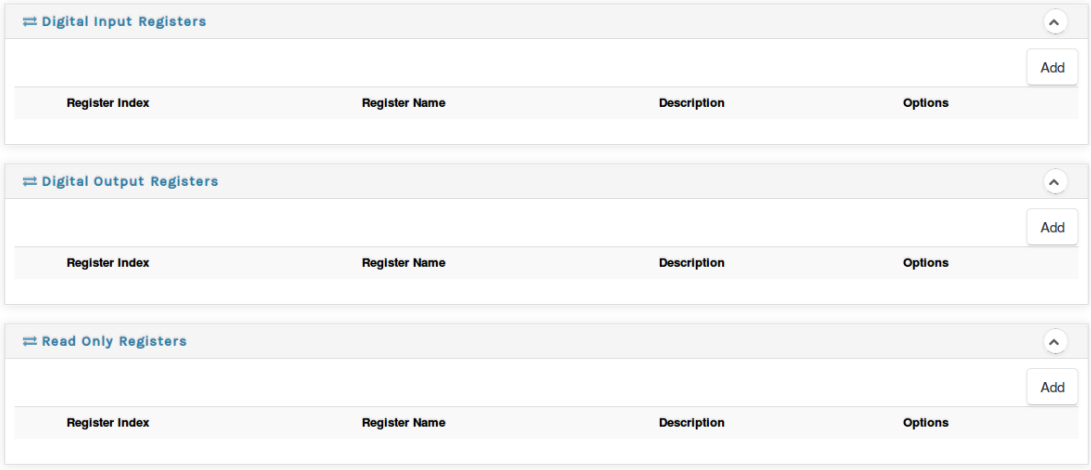
Select a unique device type ID for your device or ignore this if editing a device.
The Device Name is the short name shown in the device list.
The Device Description is a longer version of the name, keep it to less than 64 characters.
Leave the *Number of Roles, Digital Input Count, Digital Output Count, Read Only Register Count* and *Read/Write Register count* untouched – these values will automatically be modified as you add roles and registers in the sections below.

In the example above, a device driver for a Schneider Conext inverter was defined. 15 different roles for the corresponding holding registers was defined. The Read/Write registers are not in a contiguous block, so the Multiblock mode was selected. In this mode, you will need to enter the individual addresses of each register along with details.

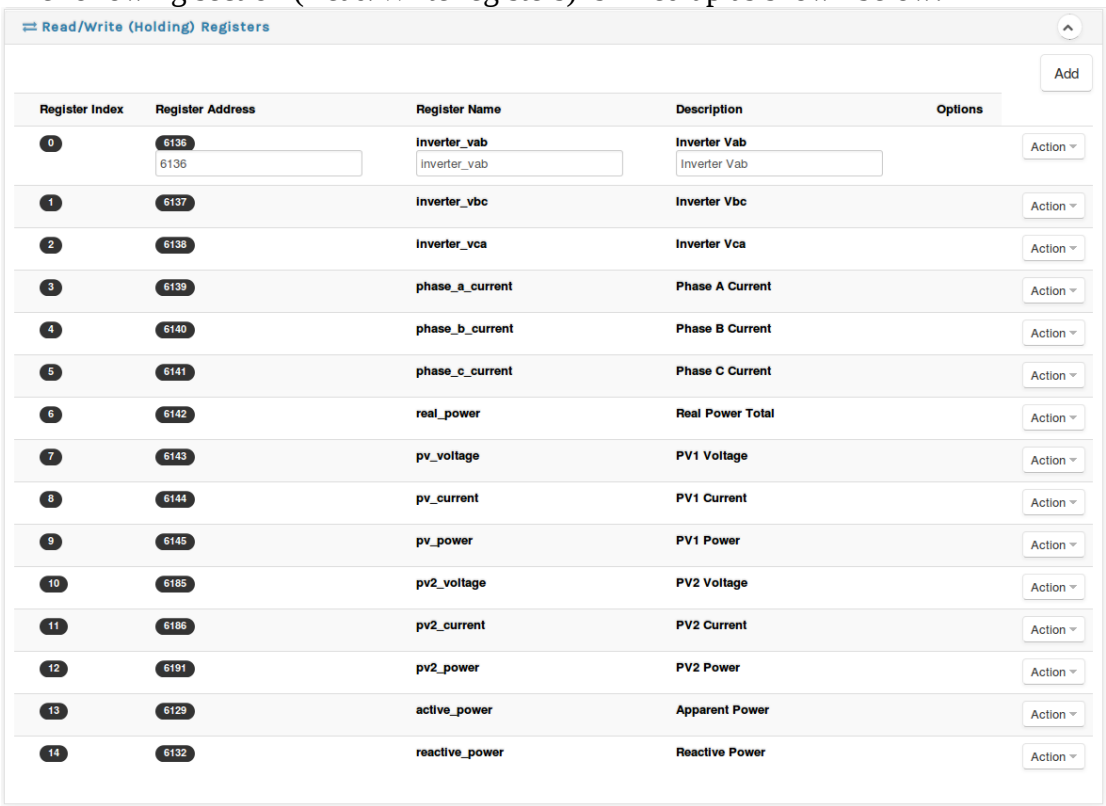
When selecting Single block mode, you just need to enter the start address of the block in the field immediately below the selection.

Adding a register

The Conext modbus interface has no digital inputs, outputs or read-only registers, hence this is left blank:



The following section (Read/Write registers) is filled up as shown below:



Add a new register by clicking the Add button. The first register (index 0) is in edit mode – this can be activated by clicking Action > Edit at the right of the row.
The Register Address corresponds to the documentation provided by the manufacturer.

The Register name is a variable name to help you identify the variable. The Description is used when assigning a role to the register in the Device Manager.

Application Note - Using Modbus With Conext™ TL Inverters

Table 30 Modbus map (Continued)

Address (hex)	Description	Access	Units	Size
0x1702	Temperature 2, Boost module 1	Read Only	C/X10	INT16
0x1703	Temperature 3, Boost module 2	Read Only	C/X10	INT16
0x1704	Temperature 4, Inverter module	Read Only	C/X10	INT16
0x17F1	Apparent Power	Read Only	kVA/x10	UINT16
0x17F4	Reactive Power	Read Only	kVAr/X10	INT16
0x17F8	Inverter Vab	Read Only	Vrms/X10	INT16
0x17F9	Inverter Vbc	Read Only	Vrms/X10	INT16
0x17FA	Inverter Vca	Read Only	Vrms/X10	INT16
0x17FB	Phase A Current	Read Only	Arms/X10	INT16
0x17FC	Phase B Current	Read Only	Arms/X10	INT16
0x17FD	Phase C Current	Read Only	Arms/X10	INT16
0x17FE	Real Power (Total)	Read Only	kW/X10	INT16
0x17FF	PV1 Voltage	Read Only	V/X10	INT16
0x1800	PV1 Current	Read Only	A/X10	INT16
0x1801	PV1 Power	Read Only	kW/X10	INT16

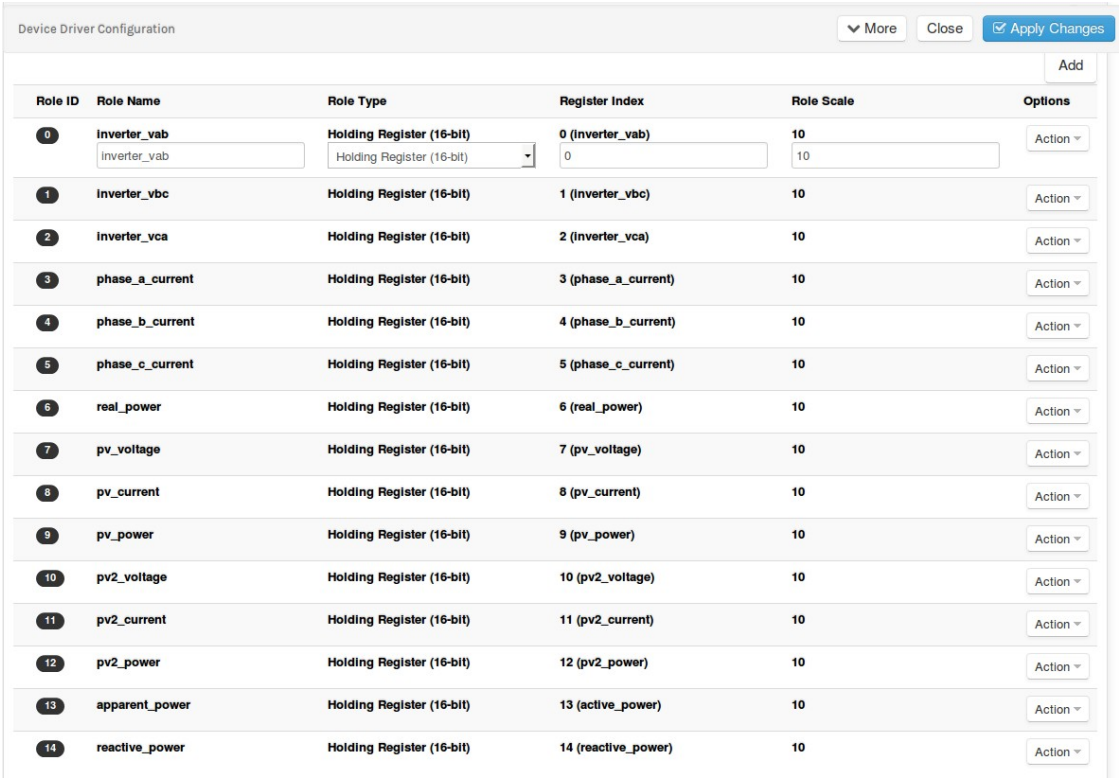
Datasheet provided by manufacturer

As shown above, all this data is taken from the manufacturer data sheet. In this case all registers need to be divided by 10 to get the actual value – this will be done in the next section.

Once the registers are entered, proceed to the roles section and fill it in.

Creating Device Roles

Device roles are the variables that glue the device driver to the Wattmon system. These need to be linked in the Device Manager after adding the device to the system. Roles are variables linked to specific registers in the modbus driver.



Role ID	Role Name	Role Type	Register Index	Role Scale	Options
0	inverter_vab	Holding Register (16-bit)	0 (inverter_vab)	10	Action
1	inverter_vbc	Holding Register (16-bit)	1 (inverter_vbc)	10	Action
2	inverter_vca	Holding Register (16-bit)	2 (inverter_vca)	10	Action
3	phase_a_current	Holding Register (16-bit)	3 (phase_a_current)	10	Action
4	phase_b_current	Holding Register (16-bit)	4 (phase_b_current)	10	Action
5	phase_c_current	Holding Register (16-bit)	5 (phase_c_current)	10	Action
6	real_power	Holding Register (16-bit)	6 (real_power)	10	Action
7	pv_voltage	Holding Register (16-bit)	7 (pv_voltage)	10	Action
8	pv_current	Holding Register (16-bit)	8 (pv_current)	10	Action
9	pv_power	Holding Register (16-bit)	9 (pv_power)	10	Action
10	pv2_voltage	Holding Register (16-bit)	10 (pv2_voltage)	10	Action
11	pv2_current	Holding Register (16-bit)	11 (pv2_current)	10	Action
12	pv2_power	Holding Register (16-bit)	12 (pv2_power)	10	Action
13	apparent_power	Holding Register (16-bit)	13 (active_power)	10	Action
14	reactive_power	Holding Register (16-bit)	14 (reactive_power)	10	Action

Roles for Conext TL driver

Add a role by clicking the *Add* button. The image above shows the Schneider Conext roles defined. Roles can link to digital inputs, digital outputs, read only and holding registers. In the role name, give the role a unique name (use _ instead of spaces in the name). For special register types spanning multiple physical registers, select the role type. The available role types are:

Digital Input

This will require you to enter the index to the register in the Digital Input section. Select a scale value of 1 for no scaling, or enter a multiplier to divide the value by the multipler each time.

Digital output

This will point to the register index in the Digital Outputs section.

Read Only (16-bit)

In this case register_index will need to be set to the index of the register in the read only section. Set the scale value appropriately (in the example above a scale of 10 signified that the raw value will be divided by 10. If the voltage is 2305, the role value will be 230.5V)

Read Only Float (32-bit)

In this case register_index will need to be set to the index of the register in the read only section. This will automatically combine the register and the followin register into a 32-bit float. This assumes the float is big-endian.

Read Only Float Little Endian (32-bit)

In this case register_index will need to be set to the index of the register in the read only section. This will automatically combine the register and the followin register into a 32-bit float. This assumes the float is little-endian.

Read Only Unsigned Long Little Endian (32-bit)

In this case register_index will need to be set to the index of the register in the read only section. This will automatically combine the register and the following register into a 32-bit unsigned int – the first register (for example index 5) would be the lower value while index 6 would be the higher value.

Read Only Unsigned Long Big Endian (32-bit)

In this case register_index will need to be set to the index of the register in the read only section. This will automatically combine the register and the following register into a 32-bit unsigned int – the first register (for example index 5) would be the higher value while index 6 would be the lower value.

Read Only Signed Long Little Endian (32-bit)

In this case register_index will need to be set to the index of the register in the read only section. This will automatically combine the register and the following register into a 32-bit signed int – the first register (for example index 5) would be the lower value while index 6 would be the higher value.

Read Only Signed Long Big Endian (32-bit)

In this case register_index will need to be set to the index of the register in the read only section. This will automatically combine the register and the following register into a 32-bit signed int – the first register (for example index 5) would be the higher value while index 6 would be the lower value.

Holding Register (16-bit)

In this case register_index will need to be set to the index of the register in the read/write section. Set the scale value appropriately (in the example above a scale of 10 signified that the raw value will be divided by 10. If the voltage is 2305, the role value will be 230.5V)

Read/Write Float Big Endian (32-bit)

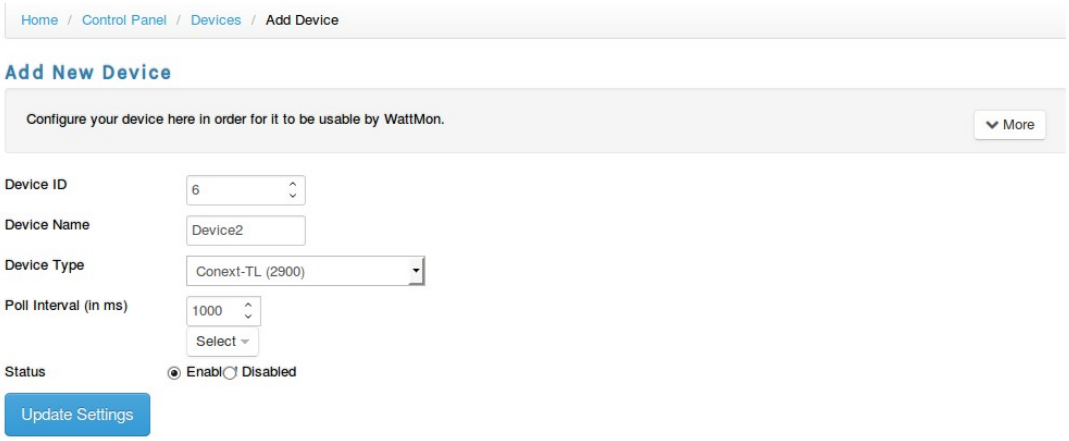
In this case register_index will need to be set to the index of the register in the read/write section. The registers are arranged as big-endian.

Read Write Register – UINT 32 bit big endian

In this case register_index will need to be set to the index of the register in the read/write section. This will create a single unsigned int from the register and the following register.

Adding the device to Wattmon

Once the device driver has been created and connected up physically (make sure each device has its own unique modbus ID), you will need to add it to Wattmon. To do so, go to the Devices page, and click Add > Manually Add a Device.



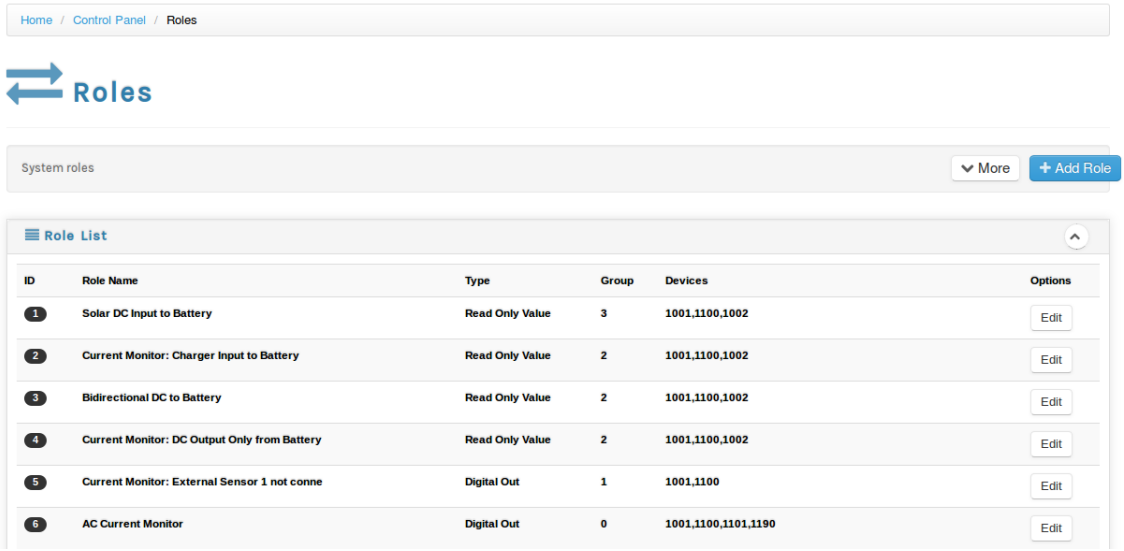
Add a device page

Select the device ID of the device you wish to connect to. Enter a unique device name or leave the default name. Select the device driver you just created from the list of drivers. Click Update Settings to add the device to the list. If the device is properly functioning and your driver is set up right, you will see the Active status next to the device in the list. Otherwise it will show Error in red. You will then need to debug this to figure out where the problem lies.

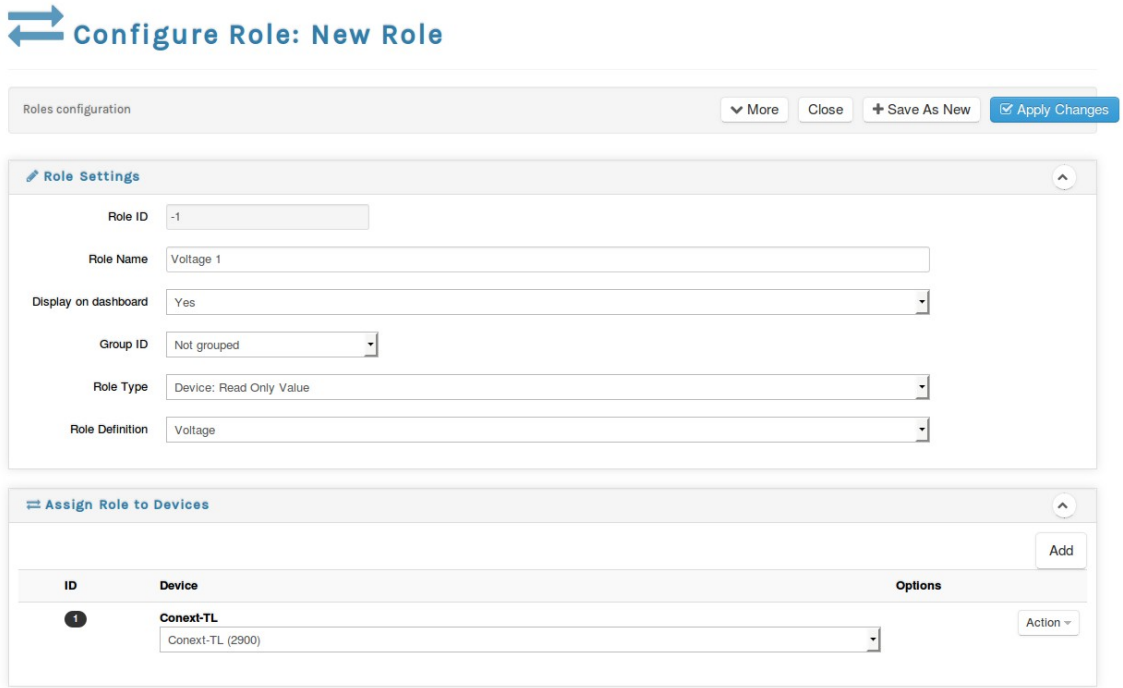
Device is active

Creating and assigning Roles

The final part of the device configuration involved creating or linking existing roles to your device driver. Go to Control Panel > Roles.



A list of all roles in the system is shown. Either edit a role or create a new role, and assign it to the device driver as shown in the next section:



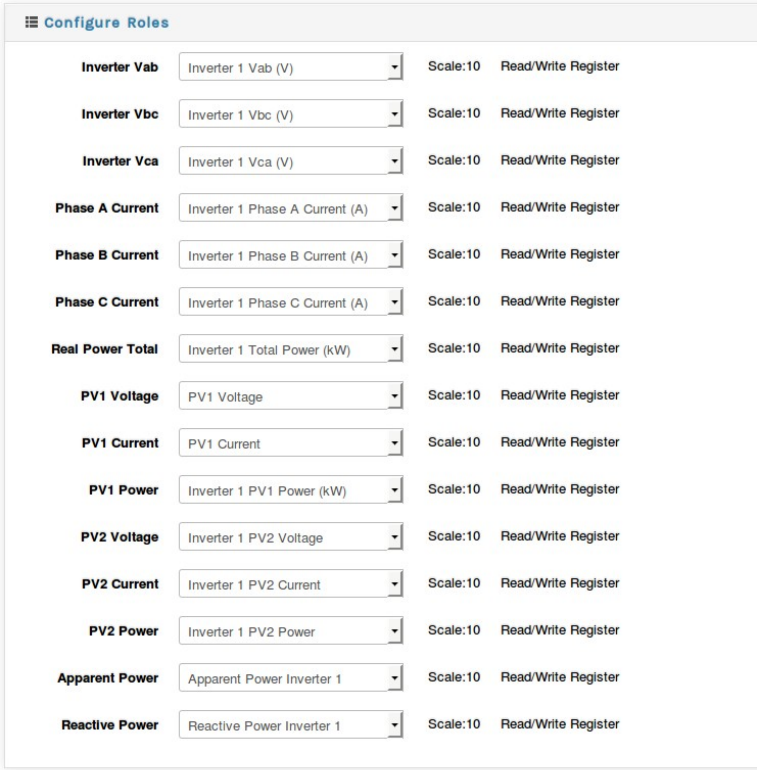
Configuring a new role

The Role name should be unique and help you to identify the purpose of the role. In this example we are creating a role called Voltage 1. If you wish the variable to be displayed in the front end on the variables widget, select Yes in Display on Dashboard. If you are defining a current, voltage or wattage, you may wish to assign it to a data group in the Group Id field. For example, if you have multiple inverters feeding solar energy into the grid, and they all have a Wattage field indicating the energy being exported, you could assign the role to the Solar group. The sum of all watt energy will be consolidated and logged as total solar energy. Make sure to specify Watts in the Role difinition.
In the Role Type, select the role type (this is the same as what you defined in the device driver above) – this can be Digital Output, Digital Input, Read Only or Read/write for modbus devices. If you are creating a onewire device role (temperature sensor), select OneWire. If you are linking a global variable to a role, select Global variable.

In the section below, assign which devices can be linked to this role in the device edit page. You can assign one role to multiple device types.

Configuring the Device

Once you have created your roles, go back to the devices page and edit the device. You will now be able to assign the roles you created to the device.



Configure the roles for the device

Now the device is fully operational. If you wish to log some or all of the values, you will need to add the roles to the data logs in the Data Collection page in Control Panel.

Application Note: Modbus Device Driver

For more information, visit http://www.wattmon.com
or email: wattmonsolar@gmail.com