ETPORT

MODBUS INTERNET GATEWAY

FEATURES:

- ♦ 10/100Mbps Ethernet and 2.4GHz WiFi b/g/n
- ♦ Modbus/TCP to Modbus/RTU bridge (multiple device support)
- ♦ Real-time monitoring via dynamic Web interface
- ♦ User defined "Virtual Datapoints" with register math
- ♦ API for Web and custom development / branding
- ♦ Push data (HTTP Post) to Elkor Cloud or custom server
- ♦ RS-485 Pass-through Port
- ♦ Responsive web interface supports desktop and mobile browsers
- ♦ Small size, DIN or wall mount. Simple, Plug-and-play installation

APPLICATIONS:

- ◊ Ethernet Accessory for WattsOn-Mark II
- ◊ Modbus/TCP to multiple Modbus/RTU slave devices
- ◊ Modbus/RTU data aggregator
- ◊ Cloud gateway for multiple Modbus slaves

SPECIFICATIONS:

Power:	9-32 VDC (100mA) or 20-24VAC (100mA)
RS-485:	Modbus/RTU
Ethernet:	10/100Mbit, full/half duplex, auto negotiation, DHCP / AUTOIP Software for detection/configuration.
WiFi:	Wi-Fi: 802.11 b/g/n
Posting:	HTTP(s), HTTP Authentication, fully customizable template format (XML, JSON, CSV, plain text, etc.)
Mounting:	DIN Rail Mount (LxWxH = 54mm x 91mm x 60mm)

PRODUCT DESCRIPTION:

The ETPort module is a Modbus/RTU to Ethernet & WiFi gateway featuring multiple simultaneous modes of operation. It is an ideal accessory for WattsOn-Mark II devices, as well as any other RS-485, Modbus/RTU slaves.

The ETPort allows instantaneous, real-time register access via its on-board web interface, transparent Modbus/TCP to Modbus/RTU bridging, and the ability to post Modbus Register data to cloud servers via a highly flexibly template structure.

Utilizing the on-board JumpBus header, the ETPort may be instantly connected to the WattsOn-Mark II, sharing the power supply and making data connections to the meter. This relieves the need for additional wiring.

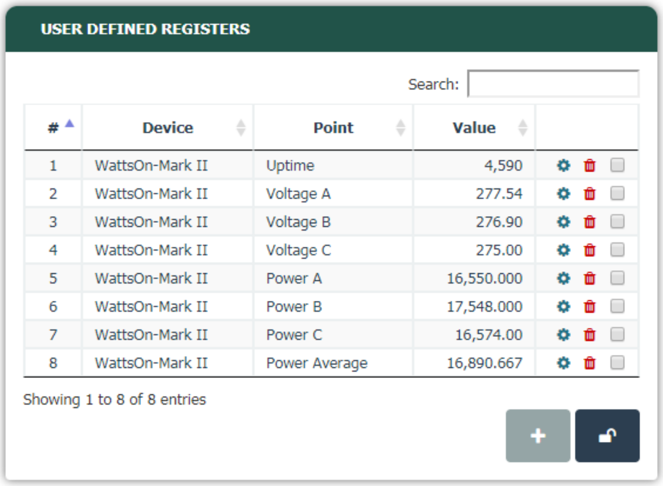
The ETPort provides a dynamic AJAX-enabled web interface. It allows easy addition of multiple Modbus/RTU devices and registers, as well as virtual points which allow the user to specify math operations (add, subtract, multiply, divide) on multiple device registers. The ETPort may send device data to a web server via HTTP(S) POST at regular intervals, using a highly configurable template-based format.

The additional RS-485 terminal provides a pass-through interface allowing for simultaneous communication from RS-485 Modbus/RTU Masters and Modbus/TCP clients via Ethernet or WiFi.

ORDERING INFORMATION:

`ETPort` fully describes this product

ELKOR



USER DEFINED REGISTERS

Search:

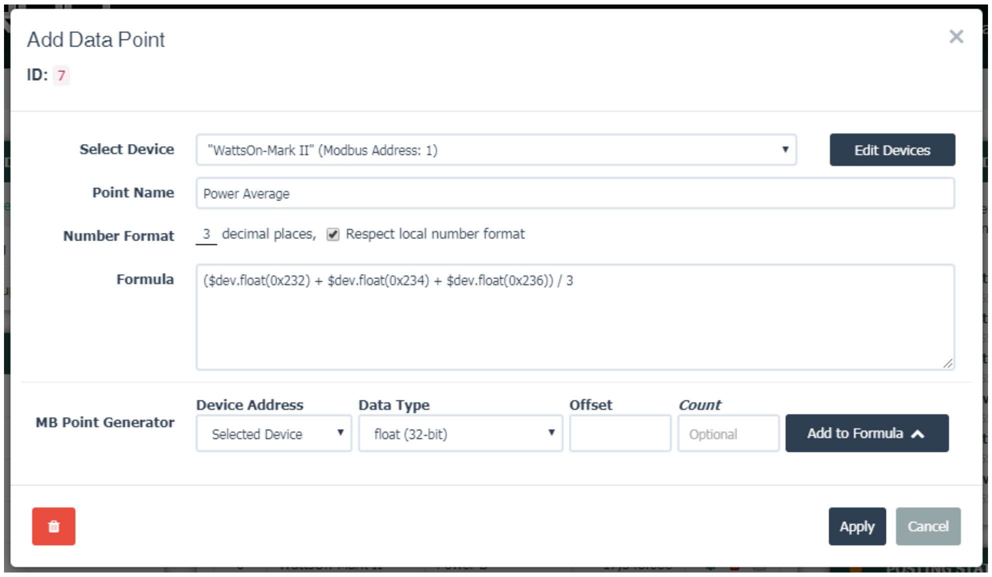
#	Device	Point	Value
1	WattsOn-Mark II	Uptime	4,590
2	WattsOn-Mark II	Voltage A	277.54
3	WattsOn-Mark II	Voltage B	276.90
4	WattsOn-Mark II	Voltage C	275.00
5	WattsOn-Mark II	Power A	16,550.000
6	WattsOn-Mark II	Power B	17,548.000
7	WattsOn-Mark II	Power C	16,574.00
8	WattsOn-Mark II	Power Average	16,890.667

Showing 1 to 8 of 8 entries

Fully customizable register listing allows users to freely add multiple Modbus devices and registers. Supports floating point, short, integer, & string datatypes.

Data points may be “virtual” data points, including arithmetic consisting of multiple other Modbus registers. This allows users to add points which add, subtract, multiply and divide by other Modbus registers or fixed constants.

An intuitive user interface allows simple management of data points and their properties.



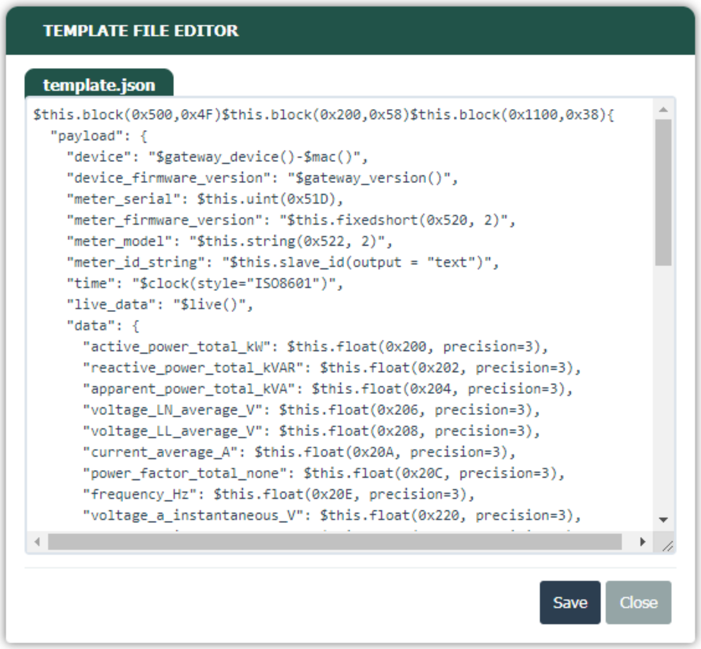
Add Data Point

ID: 7

Select Device	"WattsOn-Mark II" (Modbus Address: 1)
Point Name	Power Average
Number Format	3 decimal places, ☑ Respect local number format
Formula	($dev.float(0x232) + $dev.float(0x234) + $dev.float(0x236)) / 3

MB Point Generator

Device Address	Data Type	Offset	Count
Selected Device	float (32-bit)		Optional



TEMPLATE FILE EDITOR

template.json

```
$this.block(0x500,0x4F)$this.block(0x200,0x58)$this.block(0x1100,0x38){
  "payload": {
    "device": "$gateway_device()-$mac()",
    "device_firmware_version": "$gateway_version()",
    "meter_serial": $this.uint(0x51D),
    "meter_firmware_version": "$this.fixedshort(0x520, 2)",
    "meter_model": "$this.string(0x522, 2)",
    "meter_id_string": "$this.slave_id(output = "text")",
    "time": "$clock(style="ISO8601")",
    "live_data": "$live()",
    "data": {
      "active_power_total_kW": $this.float(0x200, precision=3),
      "reactive_power_total_kVAR": $this.float(0x202, precision=3),
      "apparent_power_total_kVA": $this.float(0x204, precision=3),
      "voltage_LN_average_V": $this.float(0x206, precision=3),
      "voltage_LL_average_V": $this.float(0x208, precision=3),
      "current_average_A": $this.float(0x20A, precision=3),
      "power_factor_total_none": $this.float(0x20C, precision=3),
      "frequency_Hz": $this.float(0x20E, precision=3),
      "voltage_a_instantaneous_V": $this.float(0x220, precision=3),
```

The flexible HTTP POST template structure allows users to define any device and registers within the structure of the template.

The text file templates may be defined in any format including text, CSV, JSON, XML, with tags used as placeholders for register data. The tags are replaced by live data when it comes time to post.

The posts are buffered internally in the event of a network outage.