HUB'O:

MODBUS INTERFACE

NOTICE

Nke Watteco reserves the right to make changes to specifications and product descriptions or to discontinue any product or service without notice. Except as provided in Nke Watteco's Standard Terms and Conditions of Sale for products, Nke Watteco makes no warranty, representation or guarantee regarding the suitability of its products for any particular application nor does Nke Watteco assume any liability arising out of the application or use of any product and specifically disclaims any and all liability, including consequential or incidental damages.

Certain applications using semiconductor products may involve potential risks of death, personal injury or severe property or environmental damage. Nke Watteco products are not designed, authorized or warranted to be suitable for use in life saving or life support devices or systems. Inclusion of Nke Watteco products in such applications is understood to be fully at the Customer's risk.

In order to minimize risks associated with the customer's application, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

Nke Watteco assumes no liability for applications assistance or customer product design. Nke Watteco does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of Nke Watteco covering or relating to any combination, machine or process in which such semiconductor products or services might be or are used. Nke Watteco's publication of information regarding any third party's products or services does not constitute Nke Watteco's approval, warranty and endorsement thereof.

Resale of Nke Watteco's products with statements of functionality different from or beyond the parameters stated by Nke Watteco for that product as defined by Nke Watteco's unique part number, voids all express and any implied warranties for that product, is considered by Nke Watteco to be an unfair and deceptive business practice and Nke Watteco is not responsible nor liable for any such use.

Embedded software is based on Nke Watteco proprietary drivers and applicative code and operates on the Contiki kernel from the SICS (Swedish Institute of Computer Science).

http://www.nke-watteco.com/

DOCUMENT HISTORY

Date	Revision	Modification Description
February 2019	1.0	Creation

CONTENTS

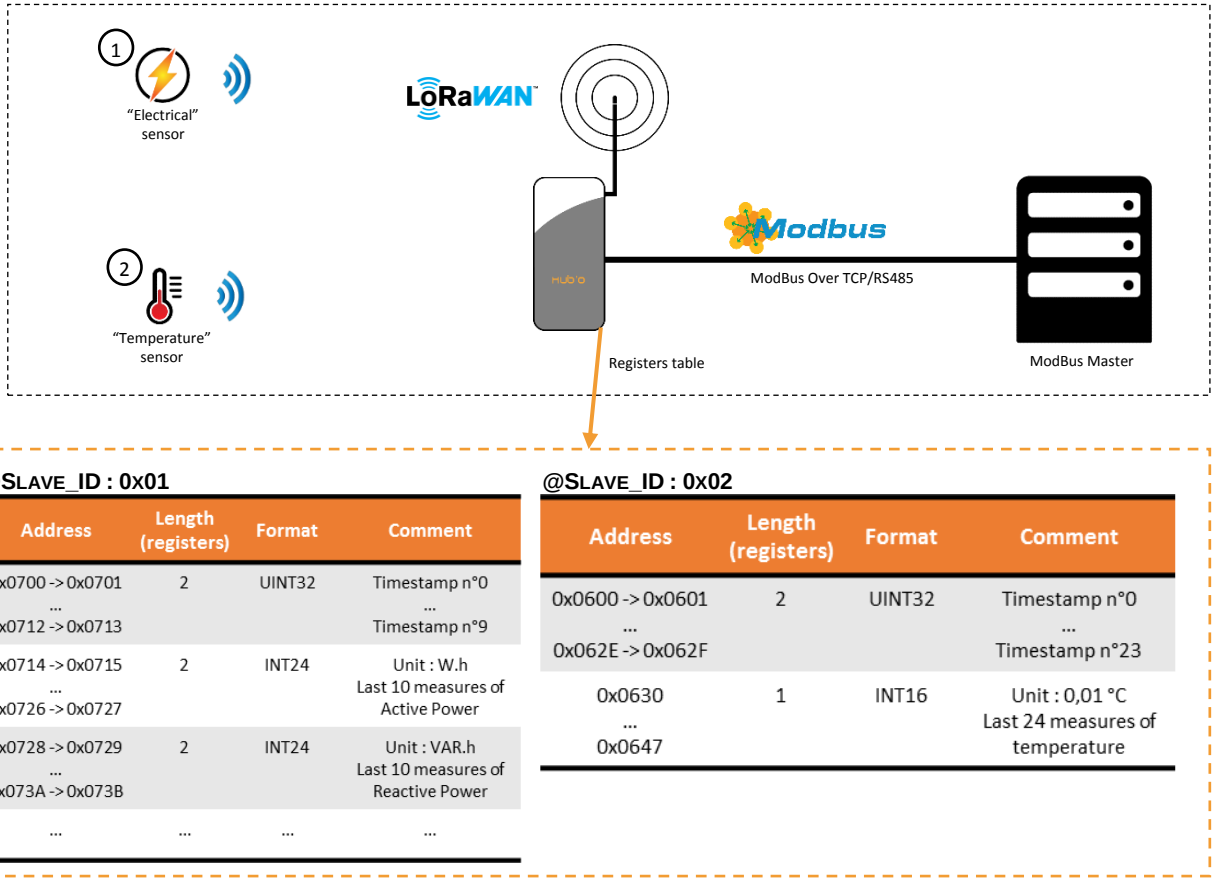
1 INTRODUCTION

With the firmware version 02.00, Hub'O offers to the user, a ModBus interface, on TCP or on a RS485 bus. This interface allows especially to interface Hub'O with a programmable logic controller.

Therefore, it allows a ModBus master to get the data received from the LoRaWAN sensors (directly decoded for most of the nke LoRaWAN sensors). It allows as well to send downlink frames to the LoRaWAN sensors, to set/get the current time used by Hub'O, to get the "Low back-up battery" alarm state or to get the last SNR, RSSI for each sensor.

Thus, this document firstly gives a general description about how works the ModBus interface on Hub'O. Then, it gives a map representation of how Hub'O "virtualizes" each LoRaWAN sensors saved as ModBus slaves. Finally, the biggest part of this document gives all the details about the different registers available on the ModBus interface.

2 GENERAL DESCRIPTION



@SLAVE_ID : 0x01

Address	Length (registers)	Format	Comment
0x0700 -> 0x0701 ... 0x0712 -> 0x0713	2	UINT32	Timestamp n°0 ... Timestamp n°9
0x0714 -> 0x0715 ... 0x0726 -> 0x0727	2	INT24	Unit : W.h Last 10 measures of Active Power
0x0728 -> 0x0729 ... 0x073A -> 0x073B	2	INT24	Unit : VAR.h Last 10 measures of Reactive Power
...	...	...	...

@SLAVE_ID : 0x02

Address	Length (registers)	Format	Comment
0x0600 -> 0x0601 ... 0x062E -> 0x062F	2	UINT32	Timestamp n°0 ... Timestamp n°23
0x0630 ... 0x0647	1	INT16	Unit : 0,01 °C Last 24 measures of temperature

FIGURE 1 - DIAGRAM OF THE MODBUS INTERFACE IN USE

The ModBus interface is an additional functionality, which can work simultaneously with the standard Hub'O behavior.

This interface is available on Hub'O over the TCP/IP protocol (with an Ethernet cable) or over a RS485 bus. All the parameters related to the TCP/IP or serial communication can be changed thanks to the modbus interface configuration file (***c_modbus_010_0000.json***). This file can be uploaded on Hub'O from the distant http server or with a USB stick.
Please see the "**HubO_Server_Exchanges_Description_X_X.pdf**" document for more information about this file.

The ModBus interface implemented on Hub'O is RTU type uniquely, not ASCII.

On one side, the LoRaWAN sensors communicates with Hub'O. The latter collects the sensors payloads and decodes it for some of them. Finally, it makes the result available on the other side: the ModBus interface.

Hub'O slaves address is 0x64 (100) by default. Moreover, for each LoRaWAN sensors communicating with Hub'O, it emulates a new slave address. The sensors slave addresses follow the Hub'O slave address. For example if, like in the diagram, two sensors are paired, their addresses will be 0x65 (101)

and 0x66 (102). If one of the sensor is deleted afterwards, Hub'O does not shift all the other addresses to keep all the addresses next to each other. Instead, it will keep the new free address and will allocate it to the next new sensor.

As a reminder, in the ModBus protocol, valid addresses are from 1 (0x01) to 247 (0xF7). Therefore, if Hub'O ModBus address is changed and it leads to end-devices ModBus addresses being higher than the maximum value (247), then those values will be shifted to the first valid ModBus address. For example, if Hub'O address is put to 245 and there are 5 end-devices paired with Hub'O, then these devices have the addresses: 246, 247, 1, 2 and 3.

In order to know which slave address corresponds to a given end-device, Hub'O makes available a "link table", where can be found the number of end-devices paired with Hub'O, the ModBus slave addresses of every paired end-devices and the corresponding devEUI. This table of registers is available on Hub'O slave address (cf. **§4.1.1**). As well as the current time used by Hub'O (cf. **§4.1.2**), or the "Low back-up battery" alarm state (cf. **§4.1.3**).

From each sensor ModBus address, can be asked the data received from this sensor and the SNR/RSSI seen during the last communication (cf. **§4.2.1**). Some registers are also available to be written: they allow to send a downlink frame to the concerned end-device (cf. **§4.2.2**).

About the data:

- For data received from nke Watteco end-devices: most of the data from our sensors are decoded by Hub'O, and are available in ModBus registers, in pair with the Timestamp of reception. For the non-decoded data, they are available in the "raw payload" registers.

- For data received from end-devices other than nke Watteco: the data are not decoded by Hub'O and are only available in the "raw payload" registers.

All the registers that can be accessed by the ModBus master for the end-devices are detailed in paragraphs **§4.2.**

All the details about the registers containing the end-devices data are given in the paragraph **§4.2.3** and **§4.2.4** of this document.

3 MAP OF THE MODBUS INTERFACE

On Hub'O ModBus interface, a lot of actions can be done (reading or writing) and a lot of registers are available. Therefore, it can be easy to get lost. To avoid that, here below can be found a global "map" of this interface.

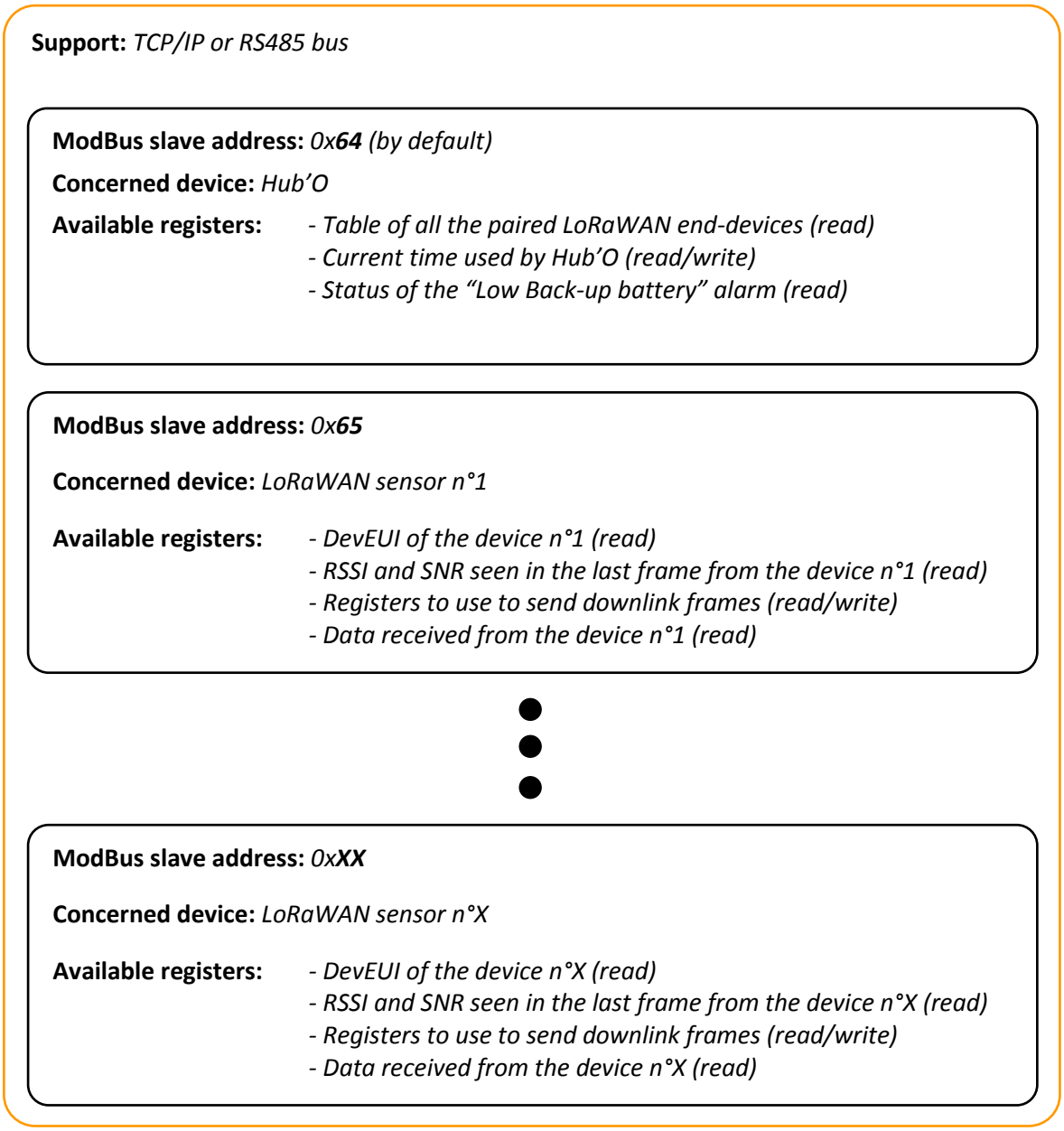


FIGURE 2 - MAP OF THE MODBUS INTERFACE

As it can be seen on the "map" here below, the ModBus interface on Hub'O provides the user with several slave ModBus addresses: one that contains the data relative to Hub'O and the other one that contain the specific data for each LoRaWAN end-device paired to Hub'O.

4 AVAILABLE REGISTERS IN DETAILS

4.1 HUB'O REGISTERS

The following registers are available only on the ModBus slave address corresponding to Hub'O (0x64 or 100, in decimal, by default).

4.1.1 LINK TABLE BETWEEN SENSORS DEVEUI AND MODBUS SLAVE ADDRESSES

@Slave_ID: 0x64 (by default)

Address	Length (registers)	Format	Operation available	Comment
0x0000	1	UINT16	Read	Number of devices in the list (so, associated to Hub'O)
0x0001 … 0x0064	1	UINT16	Read	ModBus address of every LoRaWAN end-device saved on Hub'O (address to use to read the end-device data)
0x0065 -> 0x0068 … 0x01F1 -> 0x01F4	4	HEX (8B)	Read	Devices list: devEUIs of every end-device associated to Hub'O. The ModBus address associated is at the same index in the previous row

This table of registers contains the number of LoRaWAN end-devices paired with Hub'O, the list of the ModBus slave addresses for these end-devices and the list of the corresponding devEUI.

For example, the ModBus slave address at index 5 of the first list (from register 0x0001) corresponds to the devEUI at the index 5 of the second list (from register 0x0065).

4.1.2 TIME USED BY HUB'O

@Slave_ID: 0x64 (by default)

Address	Length (registers)	Format	Operations available	Comment
0x01F5 -> 0x01F6	2	UINT32	Read/Write	Current Hub'O Time

The registers 0x01F5 and 0x01F6 contain the time (Date and hour) used by Hub'O to Timestamp the data received from the sensors. The time is stored as a long integer (on 4 bytes). It can be read or written by the ModBus master.

4.1.3 LOW BACK-UP BATTERY ALARM

@Slave_ID: 0x64 (by default)

Address	Length (registers)	Format	Operation available	Comment
0x01F7	1	BOOL	Read	Status of the Alarm : LOW BACKUP BATTERY (On (1) or Off (0))

This register contains the status of the alarm on the backup battery voltage. If the value of the register is at 1, the battery voltage is too low, and a new one is needed. If the value is at 0, the battery voltage is high enough.

4.2 LORAWAN SENSORS REGISTERS

The following registers are available only on the ModBus slave address corresponding to LoRaWAN sensors paired with Hub'O.

4.2.1 SENSOR INFORMATION

@Slave_ID: 0xii (0xii: Sensor ModBus address. Cf. link table)

Address	Length (registers)	Format	Operation available	Comment
0x0000 -> 0x0003	4	HEX (8B)	Read	DevEUI of the end-device
0x0004	1	INT16	Read	RSSI level of the last frame received from this device (in **dBm**). 0x7FFF if no value yet.
0x0005	1	INT16	Read	SNR level of the last frame received from this device (in **dBm**). 0x7FFF if no value yet.

The devEUI of the end-device is available in the first 4 registers of the simulated sensor. On the 5th register, the RSSI seen by Hub'O during the last communication with the sensor is saved. On the 6th register, it is the SNR seen during the last transmission of the sensor.

All of these registers are in read only.

4.2.2 SEND DOWNLINK FRAMES

@Slave_ID: 0xii (0xii: Sensor ModBus address. Cf. link table)

Address	Length (registers)	Format	Operations available	Comment
0x0006	1	BOOL	Read	Status of the frame sending : ready to send (0) or sending (1)
0x0007	1	UINT8	Read	Status of the last attempt to send a frame : OK (0), KO (1) or Buffer full (2)
0x0008	1	UINT8	Write	FPort to use to send the LoRaWAN payload
0x0009	1	UINT8	Write	Length of the LoRaWAN payload to send
0x000A -> 0x0023	26	HEX (52B)	Write	LoRaWAN payload to send

This set of registers is available to allow the ModBus master to send downlink frames to LoRaWAN sensors.

The first register (**0x0006**) can be accessed in reading uniquely. It gives the current status of the sending process. The ModBus master must wait until this status value is at "Ready to send" (0) before trying to send any downlink frame.

The second register (**0x0007**) can be accessed uniquely in reading as well. It gives the status of the last attempt to send a downlink frame: **OK**, **KO** or **Buffer full**. Indeed, Hub'O can store up to 5 frames for a given end-device (specifically class A end-devices). If this number is reached, the ModBus master must wait until the given end-device communicates to allow Hub'O to send a downlink frame. Afterward, the ModBus master is be able to send a new downlink frame.

The third register (**0x0008**) can only be written by the ModBus master. It allows the latter to define the FPort to use to send the payload set in the registers 0x000A to 0x0023.

The fourth register (**0x0009**) can only be written by the ModBus master. It should contains the size (in bytes) of the payload to send.

Finally, the registers from **0x000A** to **0x0023** contain the payload that Hub'O needs to send to the given end-device. Therefore, the ModBus master needs to write it there. It is the writing on these bytes that triggers the sending (class C end-devices) or the saving (class A end-devices) of the payloads addressed to the end-device.

4.2.3 DECODED DATA REGISTERS

As it has been said at the beginning of this document, the ModBus interface decodes most of the payloads received from nke Watteco sensors. Afterwards, the data decoded are sorted in different registers, according to the cluster (temperature, humidity, binary input, etc.) used by the nke Watteco end-device.

Therefore, in this paragraph, can be found all the details about the registers containing the decoded data, sorted by cluster.

4.2.3.1 ANALOG INPUT (CLUSTER ID: 0X000C)

Available attribute for this cluster:

- **Present value** (attribute ID : **0x0055**)

Registers addresses used for this cluster: **0x0100** to **0x01FF** included.

Address	Length (registers)	Format	Operation available	Comment
0x0100 -> 0x0101 … 0x012E -> 0x012F	2	UINT32	Read	Timestamp measure n°0 (EP0) … Timestamp measure n°23 (EP0)
0x0130 -> 0x0131 … 0x015E -> 0x015F	2	FLOAT32	Read	Last 24 measures of Analog Input. EndPoint n°0
0x0160 -> 0x0161 … 0x018E -> 0x018F	2	UINT32	Read	Timestamp measure n°0 (EP1) … Timestamp measure n°23 (EP1)
0x0190 -> 0x0191 … 0x01BE -> 0x01BF	2	FLOAT32	Read	Last 24 measures of Analog Input. EndPoint n°1

4.2.3.2 BINARY INPUT (CLUSTER ID: 0X000F)

Available attributes for this cluster:

- **Present value** (attribute ID : **0x0055**)
- **Counter** (attribute ID : **0x0402**)

Registers addresses used for this cluster: **0x0200** to **0x04FF** included.

Address	Length (registers)	Format	Operation available	Comment
0x0200 -> 0x0201 … 0x0212 -> 0x0213	2	UINT32	Read	Timestamp present value n°0 (EP0) … Timestamp present value n°9 (EP0)
0x0214 … 0x021D	1	BOOL	Read	Last 10 values of Present Value. EndPoint n°0
0x021E -> 0x021F … 0x0230 -> 0x0231	2	UINT32	Read	Timestamp Counter n°0 (EP0) … Timestamp Counter n°9 (EP0)
0x0232 -> 0x0233 … 0x0244 -> 0x0245	2	UINT32	Read	Last 10 values of Counter. EndPoint n°0
…	…	…	…	…
0xrrrr -> **(0xrrrr+1)** … (**0xrrrr+18**) -> (**0xrrrr+19**)	2	UINT32	Read	Timestamp present value n°0 (EPX) … Timestamp present value n°9 (EPX)
(**0xrrrr+20**) … (**0xrrrr+29**)	1	BOOL	Read	Last 10 values of Present Value. EndPoint n°X
(0xrrrr+30) -> **(0xrrrr+31)** … (**0xrrrr+48**) -> (**0xrrrr+49**)	2	UINT32	Read	Timestamp Counter n°0 (EPX) … Timestamp Counter n°9 (EPX)
(0xrrrr+50) -> **(0xrrrr+51)** … (**0xrrrr+68**) -> (**0xrrrr+69**)	2	UINT32	Read	Last 10 values of Counter. EndPoint n°X

The ModBus interface gives access to 10 EndPoints on the "Binary Input" cluster. For each EndPoint, the last 10 "Present value" and "Counter" are available. The registers addresses for the EndPoint n°0 are given directly. For the other EndPoints, a generic value is given. It is possible to calculate the addresses by knowing that: **0xrrrr = 0x0200 + X*70**, *with X: number of the EndPoint*

For example, the registers addresses containing the data for EndPoint n°9 are from register **0x0476** to register **0x04BB**.

4.2.3.3 Relative Humidity (cluster ID: 0x0405)

Available attribute for this cluster:

- **Measured value** (attribute ID : **0x0000**)

Registers addresses used for this cluster: **0x0500** to **0x05FF** included.

Address	Length (registers)	Format	Operation available	Comment
0x0500 -> 0x0501 ... 0x052E -> 0x052F	2	UINT32	Read	Timestamp measure n°0 (EP0) ... Timestamp measure n°23 (EP0)
0x0530 ... 0x0547	1	UINT16	Read	Unit : 0.01% Last 24 measures of Humidity. EndPoint n°0
0x0548 -> 0x0549 ... 0x0576 -> 0x0577	2	UINT32	Read	Timestamp measure n°0 (EP1) ... Timestamp measure n°23 (EP1)
0x0578 ... 0x058F	1	UINT16	Read	Unit : 0.01% Last 24 measures of Humidity. EndPoint n°1

4.2.3.4 Temperature Measurement (cluster ID: 0x0402)

Available attribute for this cluster:

- **Measured value** (attribute ID : **0x0000**)

Registers addresses used for this cluster: **0x0600** to **0x06FF** included.

Address	Length (registers)	Format	Operation available	Comment
0x0600 -> 0x0601 ... 0x062E -> 0x062F	2	UINT32	Read	Timestamp measure n°0 (EP0) ... Timestamp measure n°23 (EP0)
0x0630 ... 0x0647	1	INT16	Read	Unit : 0.01°C Last 24 measures of Temperature. EndPoint n°0
0x0648 -> 0x0649 ... 0x0676 -> 0x0677	2	UINT32	Read	Timestamp measure n°0 (EP1) ... Timestamp measure n°23 (EP1)
0x0678 ... 0x068F	1	INT16	Read	Unit : 0.01°C Last 24 measures of Temperature. EndPoint n°1

4.2.3.5 SIMPLE METERING (CLUSTER ID: 0X0052)

Available attribute for this cluster:

- **Current Metering** (attribute ID : **0x0000**)
  - Contains : Active Energy, Reactive energy, active power and reactive power

Registers addresses used for this cluster: **0x0700** to **0x07FF** included.

Address	Length (registers)	Format	Operation available	Comment
0x0700 -> 0x0701 … 0x0712 -> 0x0713	2	UINT32	Read	Timestamp measure n°0 (EP0) … Timestamp measure n°9 (EP0)
0x0714 -> 0x0715 … 0x0726 -> 0x0727	2	INT24	Read	Unit : W.h Last 10 measures of Active Energy EndPoint n°0
0x0728 -> 0x0729 … 0x073A -> 0x073B	2	INT24	Read	Unit : VAR.h Last 10 measures of Reactive Energy EndPoint n°0
0x073C … 0x0745	1	INT16	Read	Unit : W Last 10 measures of Active Power EndPoint n°0
0x0746 … 0x074F	1	INT16	Read	Unit : VAR Last 10 measures of Reactive Power EndPoint n°0
0x0750 -> 0x0751 … 0x0762 -> 0x0763	2	UINT32	Read	Timestamp measure n°0 (EP1) … Timestamp measure n°9 (EP1)
0x0764 -> 0x0765 … 0x0776 -> 0x0777	2	INT24	Read	Unit : W.h Last 10 measures of Active Energy EndPoint n°1
0x0778 -> 0x0779 … 0x078A -> 0x078B	2	INT24	Read	Unit : VAR.h Last 10 measures of Reactive Energy EndPoint n°1
0x078C … 0x0795	1	INT16	Read	Unit : W Last 10 measures of Active Power EndPoint n°1
0x0796 … 0x079F	1	INT16	Read	Unit : VAR Last 10 measures of Reactive Power EndPoint n°1
0x07A0 -> 0x07A1 … 0x07B2 -> 0x07B3	2	UINT32	Read	Timestamp measure n°0 (EP2) … Timestamp measure n°9 (EP2)
0x07B4 -> 0x07B5	2	INT24	Read	Unit : W.h

… 0x07C6 -> 0x07C7				Last 10 measures of Active Energy EndPoint n°2
0x07C8 -> 0x07C9 … 0x07DA -> 0x07DB	2	INT24	Read	Unit : VAR.h Last 10 measures of Reactive Energy EndPoint n°2
0x07DC … 0x07E5	1	INT16	Read	Unit : W Last 10 measures of Active Power EndPoint n°2
0x07E6 … 0x07EF	1	INT16	Read	Unit : VAR Last 10 measures of Reactive Power EndPoint n°2

4.2.3.6 CONFIGURATION (CLUSTER ID: 0X0050)

Available attribute for this cluster:

- **Node Power Descriptor** (attribute ID : **0x0006**)
  - Contains : disposable battery voltage, rechargeable battery voltage, solar harvesting voltage and TIC harvesting voltage

Registers addresses used for this cluster: **0x0700** to **0x07FF** included.

Address	Length (registers)	Format	Operation available	Comment
0x0800 -> 0x0801 … 0x0812 -> 0x0813	2	UINT32	Read	Timestamp measure n°0 … Timestamp measure n°9
0x0814 … 0x081D	1	UINT16	Read	Unit : mV Last 10 measures of external power supply voltage
0x081E … 0x0827	1	UINT16	Read	Unit : mV Last 10 measures of rechargeable battery voltage
0x0828 … 0x0831	1	UINT16	Read	Unit : mV Last 10 measures of disposable battery voltage
0x0832 … 0x083B	1	UINT16	Read	Unit : mV Last 10 measures of solar harvesting voltage
0x083C … 0x0845	1	UINT16	Read	Unit : mV Last 10 measures of TIC harvesting voltage

4.2.4 RAW PAYLOADS REGISTERS

For the payloads not managed by Hub'O (from non-nke Watteco devices or contained in clusters not decoded yet), the ModBus interface makes them available as raw payloads in the following registers. A timestamp and a port is associated with each payload.

The payload corresponds to the "FrmPayload" of the LoRaWAN protocol.

Registers addresses used for the raw payloads: **0xFF00** to **0xFFFF** included.

Address	Length (registers)	Format	Operation available	Comment
0xFF00 -> 0xFF01	2	UINT32	Read	Timestamp of last payload
0xFF02	1	UINT16	Read	Applicative port of last payload
0xFF03	1	UINT16	Read	Length of last payload (bytes)
0xFF04 -> 0xFF1D	26	HEX (52B)	Read	Last applicative payload
0xFF1E -> 0xFF1F	2	UINT32	Read	Timestamp of second to last payload
0xFF20	1	UINT16	Read	Applicative port of second to last payload
0xFF21	1	UINT16	Read	Length of second to last payload (bytes)
0xFF22 -> 0xFF3B	26	HEX (52B)	Read	Second to last applicative payload