

5 The eWON IO Servers

5.1 Introduction

This introduction repeats some information already introduced in chapter “Tag definition: Introduction” on page 67.

An IO Server is the interface between a changing value and the eWON monitoring engine. It is a kind of driver. Any variable from any *data source* must have a common representation for all IO Servers in order to define common interface in the eWON.

The *data-source* representation in the eWON uses 3 fields for the definition of a Tag:

- The IO Server Name
- The Topic name
- The Item Name

A Tag's data-source will be uniquely identified with these 3 parameters:

IO Server name:	Is a kind of driver name. For each IO Server there is a specific Topic Name and Item Name syntax. Example: MODBUS, EWON, MEM
Topic Name:	Is used to group items inside an IO Server, for example the memory IO Server uses the blank topic ("") and the retentive topic ("ret"). All Tags of the MEM IO Server defined in the "ret" topic will have their value saved and restored when the eWON boots. All IO servers do not use a Topic Name. In that case the Topic Name field must be left empty.
Item Name:	The item name is a string of characters; its syntax is specific to each IO Server. The Item Name describes the physical variable to monitor that uses the IO Server.

For example, the MODBUS IO Server needs to poll registers or coils from a slave, so it uses an item name representation to define the *register type, register address and slave address*. (Example "40001,5" => Where 4 means "read write register", 0001 is the register number and 5 is the slave Modbus address).

Table 56: Tags data-source parameters

Important note: For optimisation purpose, the eWON may disable the polling of “invalid tags” (See “IO Server Init” on page 86).

5.2 IO servers setup

Some of the IO servers are configurable.

The IO setup window proposes a list of IO servers:

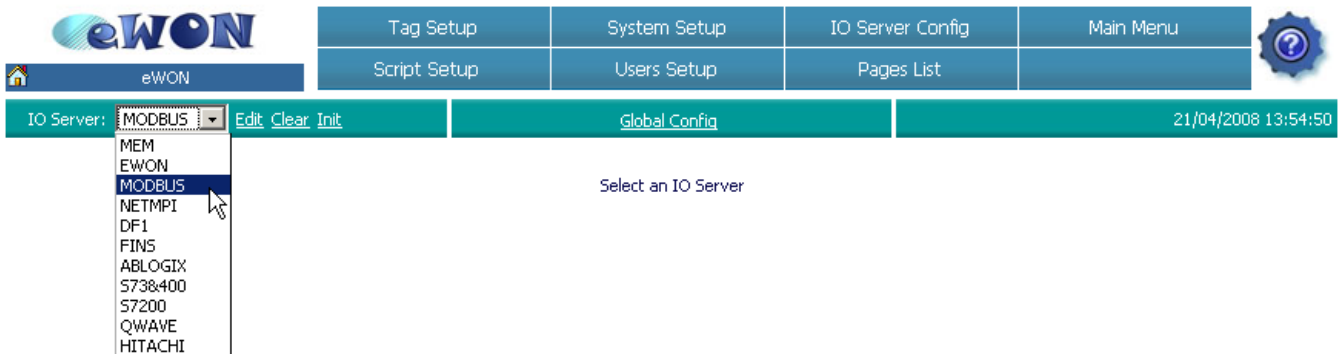


Figure 71: eWON IO servers scrolling list

Click on the **Edit** hyper link or select another IO server to display its edition window.

There are 3 possible cases regarding the IO server configuration:

- The IO server is not configurable
- The IO server has a dedicated configuration page (ex: MODBUS, UNITE, NETMPI, DF1, ...)
- The IO server uses the standard IO server configuration page.

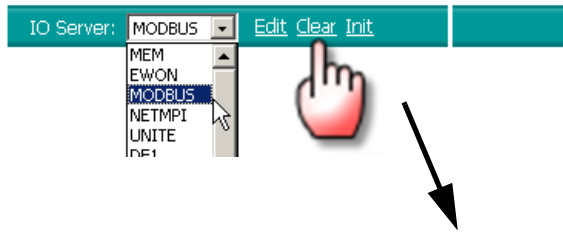
5.2.1 IO Server Clear

To avoid the useless consumption of CPU and memory, the unused IO Servers need to be “unloaded” from CPU tasks and the used memory need to be cleared.
For that purpose, the Clear function must be used.

Only set to *disable* the BaudRate of an IO Server make it idle, but this IO Server is still in memory and use some CPU time (for nothing).

Click on the **Clear** link to reset the config of the displayed IO Server and unload it from memory.

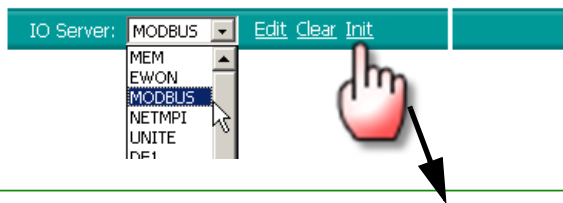
Some IO Servers are not dynamically stoppable and need an eWON reboot to ensure the unload from memory. You will be warned with the following message.



Config cleared and IO server stopped.

5.2.2 IO Server Init

Click on the **Init** link to initialise an IO Server.
This initialisation will reset all IO Server counters (See “Status” on page 159) and restart the tags validation process (See “Tags validation” on page 68).



IO Server counters reset and reinitialisation of tag polling done.

5.2.3 Global Config

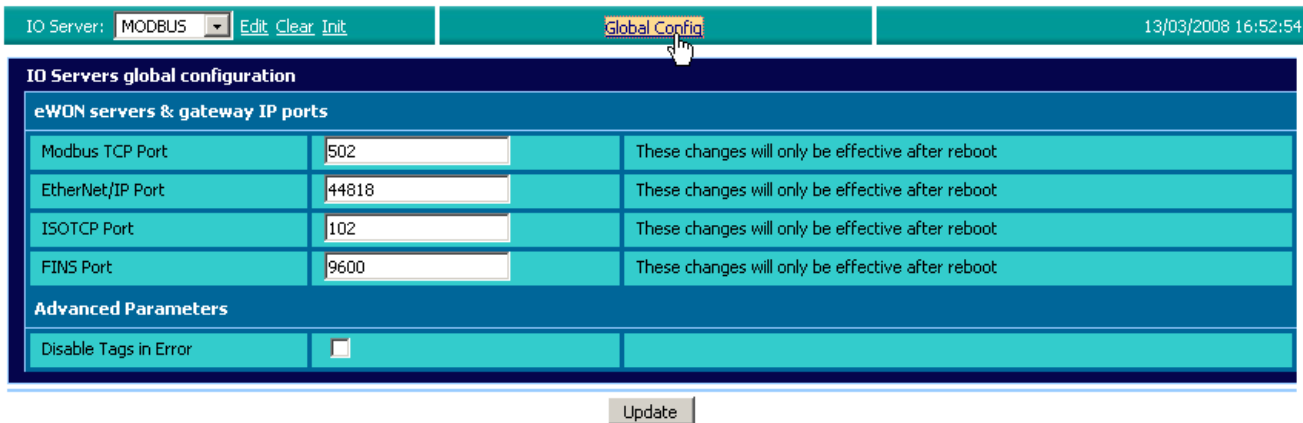
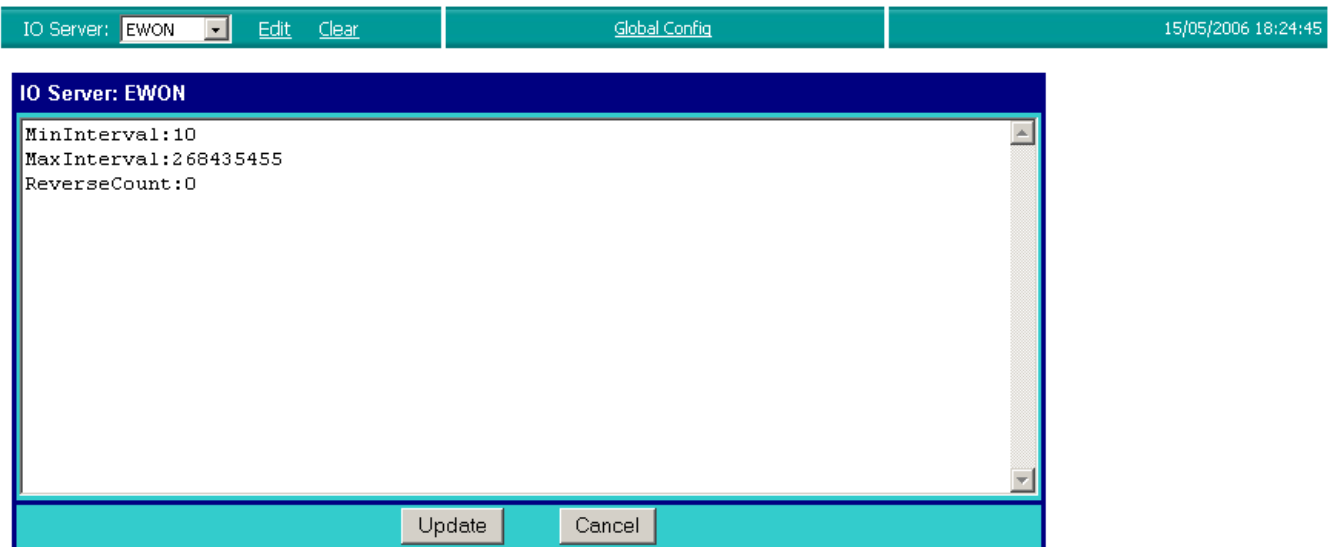


Figure 72: IO Server Global Config page

Item	Description
Modbus TCP Port	Encode here the TCP Port number used for the ModbusTCP server protocol. <i>Default is 502</i>
Ethernet/IP Port	Encode here the TCP Port used for the Ethernet/IP protocol (Allen Bradley). <i>Default is 44818</i>
ISOTCP Port	Encode here the TCP Port number used for the ISOTCP protocol (Siemens). <i>Default is 102</i>
FINS Port	Encode here the TCP Port number used for the FINS protocol (Omron). <i>Default is 9600</i>
Disable Tags in Error	Activate the Tags Validation See “Tags validation” on page 68

5.2.4 Standard IO server configuration page

When no dedicated configuration page is defined for configuring an IO server, the standard configuration page is used.



The screenshot shows a web-based configuration interface. At the top, there is a header bar with a teal background. On the left, it says "IO Server: EWON" with a dropdown arrow. Next to it are "Edit" and "Clear" buttons. In the center, it says "Global Config". On the right, it shows the date and time "15/05/2006 18:24:45". Below the header is a main configuration area with a blue border. The title of this area is "IO Server: EWON". Inside, there is a text input field containing the following text:
MinInterval:10
MaxInterval:268435455
ReverseCount:0
At the bottom of this area are two buttons: "Update" and "Cancel".

Figure 73: Standard IO server configuration page

As you can see in the above example, the standard configuration screen is a simple text edition area. Each parameter is entered on a separate line, the parameter value is separated from the parameter name by a colon ':'.

The generic format of a line is:

PARAM_NAME:PARAM_VALUE

Warning: Extra space must be removed.

When using this configuration, you must respect the correct syntax of each parameter and the values for each parameter.

The list of valid parameters and their corresponding valid values are listed in the corresponding IO server documentation (see following chapters).

Any error that would occur when the eWON applies the configuration that you have entered would be written to the event file. Please refer to chapter "Files transfer" on page 154 to see how to get access to the events file.

5.3 Modbus IO server

5.3.1 Introduction

The MODBUS IO Server setup is the standard remote IO communication setup of the eWON. It is used to configure:

- The eWON as a Modbus RTU master.
- The eWON as a Modbus TCP slave and master.

The first feature (Modbus TCP slave) is specific to the MODBUS IO server; it is actually designed to provide access to eWON Tag values and, unlike all the other IO servers, for interfacing field values with the eWON.

The second feature (MODBUS Master) is the actual IO server feature that provides an interface to the field values as a common IO server.

The eWON MODBUS IO server will give access to values in equipments having a MODBUS interface.

The interface can be:

- RS485 – MODBUS RTU protocol will be used
- ETHERNET/PPP – MODBUS TCP protocol will be used.

The eWON can mix access to MODBUS RTU and MODBUS TCP, depending on the way the Tag address is defined.

5.3.2 Setup

5.3.2.1 Setup for eWON Server

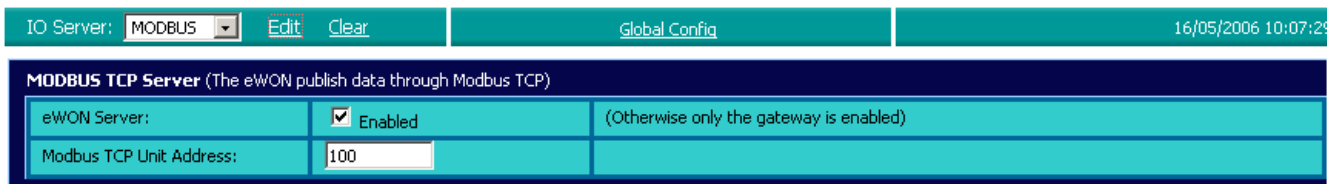


Figure 74: Modbus TCP Server configuration

This page defines the eWON configuration when used as a Modbus TCP slave.

As described in the Tag configuration paragraph, each Tag can be published to Modbus TCP so that a Modbus TCP can read their values.

This setup screen defines the eWON address, and globally enables or disables the Modbus TCP slave feature.

eWON Server Properties	Description
eWON server enabled	Globally enables or disables the Modbus TCP Server feature. If disabled, then any request from a Modbus TCP master will be rejected, even if Tags are published.
Modbus TCP Unit address	This feature is used by some gateway but can usually be left to 1 because Modbus TCP appears as a point to point connection.

Table 57: eWON server configuration - eWON as Modbus TCP slave

5.3.2.2 Setup for eWON IO server and Gateway - COM Setup

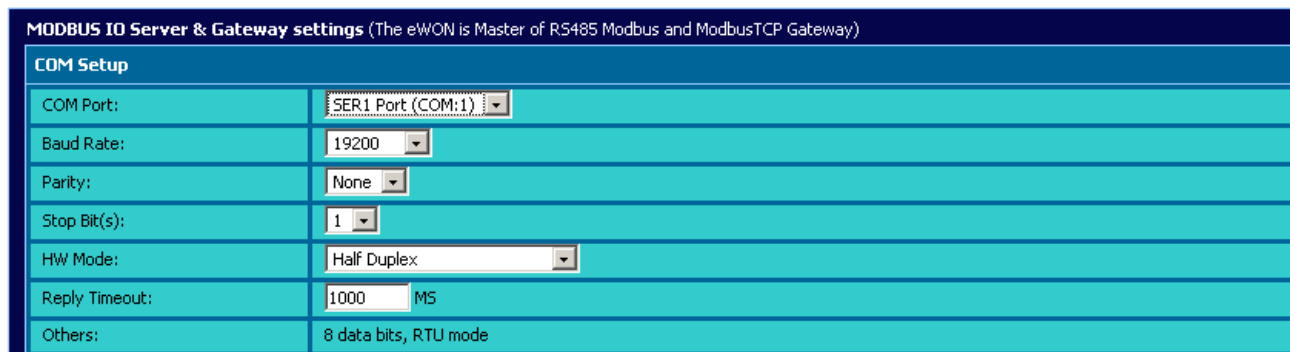


Figure 75: Modbus communications configuration

If more than one Serial port are available, you must choose on which COM the modbus request will be sent.

This configuration part defines the RS485 setup. The four first fields are used to define the baud rate, parity, number of stop bits and the reply timeout (in msec – usually 1000 msec).

Warning: When there are multiple IO servers using potentially the serial line, then the unused IO server must be Cleared or the baudrate must be set to Disabled.

Example: if Modbus and UniTE IO servers are available, at least one of them must have its baud rate set to Disabled. If not the case, one of the IO servers will not be able to use the serial line and it will be disabled, with an error written in the event log.

5.3.2.3 Topic configuration

The screenshot shows a configuration window for Modbus topics. It contains three sections, each for a different topic:

- Topic A:** Enabled. Topic Name: A. Global Slave Address: Slave Address (Unit Id): [input], IP Address (Blank for RTU): [input]. Poll Rate: 2000 MS.
- Topic B:** Enabled. Topic Name: B. Global Slave Address: Slave Address (Unit Id): [input], IP Address (Blank for RTU): [input]. Poll Rate: 2000 MS.
- Topic C:** Disabled. Topic Name: C. Global Slave Address: Slave Address (Unit Id): [input], IP Address (Blank for RTU): [input]. Poll Rate: 2000 MS.

Figure 76: Modbus topics configuration

Three topics can be used for the IO Server. These topics are used to give a common property to a group of MODBUS Tag like:

- Enable/Disable
- Poll rate
- Slave address (Modbus RTU)
- Unit address and TCP/IP address (Modbus TCP)

Modbus Server Properties	Description
Topic enabled	Enables or disables polling of all the Tags in the topic.
Slave address	The slave address is a global parameter for all the Tags of the topic. If the slave is connected with Modbus RTU, Slave address must be entered and the IP address must be blank. If the slave is Modbus TCP, its unit address and its IP address must be entered.
Poll rate	This defines the rate to which the Tag will be refreshed. In a complex application, we can imagine that some Tags must be refreshed every second – typically for digital input - and other every minute – typically: temperature-.

Table 58: Modbus topic configuration

Warning:

Any slave address that is defined in the Topic configuration overwrites the slave address configured per Tag.

If a Tag is defined with Tag Address: 40010,5 and the global address of the topic is 5 and 10.0.0.81, the Tag is entered as Modbus RTU but it is polled as Modbus TCP. So, if you need to address slaves Tag by Tag, let the topic address configuration empty.

5.3.2.4 Advanced parameters

The screenshot shows the 'Advanced parameters' configuration window. On the left, there is a text box labeled 'ComPortNum: 1'. On the right, there is a list of advanced parameters with their descriptions:

- MaxDeltaReqX:** max. number of registers grouped for reading
- MaxDeltaCoilX:** max. number of coils grouped for reading
- ErrorRetry:** max. number retry before entering slow poll mode
- SlowPollSkip:** number of scan skipped when in slow poll
- PostErrorDelay:** This delay (MSec) is added each time a device reponds with a framing error or a CRC error. Helps the bus to resynchronize
- PreDelayVal:** Delay in MSec inserted before request to a slave if previous request was sent to another slave address (not between 2 requests to the same slave).
- PreDelayFrom:** The PreDelayVal is applied only stating from this address, put this to 256 if pre delay is not used.

At the bottom of the window, there are two buttons: 'Update Config' and 'Cancel'.

Figure 77: Modbus advanced parameters configuration

Extended parameters have been added to accommodate various special operating conditions. They are entered in the edit box at the bottom of the configuration page, conforming to the syntax below. Each parameter has a default value, so the advanced parameter edit box must only be filled with the parameters for which default values must be changed. (c.f. "Standard IO server configuration page" page 87).

Parameter name	Description
PreDelayFrom	Used in conjunction with the next parameter (PreDelayVal), starting at that slave address, the eWON will insert a delay when switching from one slave address to another. If the PreDelayVal feature is not used, then the value for PreDelayFrom must be set to 256 (default value).
PreDelayVal	Used to define the delay (in Msec) to be inserted before a request to a slave if a request was previously sent to another slave address (not between 2 requests to the same slave). The PreDelayVal is placed only for slaves with an address higher than PreDelayFrom.
ErrorRetry	Defines the number of errors before the device enters in slow poll mode. (Minimum 1)
SlowPollSkip	Defines the number of times the slave is skipped when in slow poll mode. This delay depends on the poll rate.
GatewayIPCTimeout	Maximum event lock while waiting for a response to a modbus gateway request from the modbus IO server (router) (in Msec – minimum 1000).
PostErrorDelay	This delay (in Msec) is added each time a device responds with a framing error or a CRC error, in order to help the bus to perform its re synchronization (default value = 50).
MaxDeltaCoilX (X= A,B,C)	Maximum number of Coils that can be grouped in a request (per topic) max=256
MaxDeltaRegX (X= A,B,C)	Maximum number of registers that can be grouped in a request (per topic) max=124

Table 59: extended IO server configuration controls

5.3.2.4.1 Additional advanced parameters

• gwdestaddr

When the eWON is used as a Modbus gateway, it uses the UnitID from the ModbusTCP request to determine the Modbus RTU destination address.

It is possible to bypass this mechanism and force all the requests received by the eWON in ModbusTCP to be forwarded to a single ModbusRTU device (EXCEPT for requests with UnitID corresponding to the eWON's Modbus TCP Unit address (usually 100) when the eWON Server is enabled - see explanations about this precise point in the above configuration fields descriptions).

Every request is forwarded to the device with Slave address defined by the Modbus IO server advanced parameter called **gwdestaddr**.

If the advanced parameters textfield contains the following entry:

gwdestaddr:10

Then all therequests will be forwarded to the slave with address 10.

REMARK: the eWON will change the address before sending the request to the slave, then it will change it back when sending the response to the master (Modbus TCP master). So the **gwdestaddr** will never appear in any communication between the Master and the eWON.

• FastSrv

FastSrv is a mode which allows a supervisor to read in Modbus TCP more easily the Modbus tags published by the eWON. This mode completely changes the tag's addressing, and when activated, the Modbus addresses are no more compatible.

You have just to enter "FastSrv:1" in the Advanced Parameters text area to activate it. When done, the Modbus tags can be read as follows:

X	Integer (with scale factors and offset defined)
X+2048	Float (X+2048: 1 st float, X+2048+2: 2 nd float, etc.)
X+4096	Alarm status
X+6144	Alarm type

Notes:

- If the tag is binary read, its value is worth 0 if tag==0, and 1 if tag <>0
- Writing 0 in AlarmStatus acknowledges the alarm (will be logged by eWON as acknowledged by administrator)
- It is not possible to write a coil in the float area (coil address range: X+2048 to X+4094)
- It is not possible to address more than 1024 registers in float.

Click on the **Update Config** button to validate your inputs or use the **Cancel** button to discard changes

• TcpPort

Use the 'TcpPort' parameter to change the default 502 port used when the eWON is ModbusTCP CLIENT.

If not specified the 502 default value is used.

This Port value is used for all the ModbusTCP client connections.

5.3.3 Tag name convention

IO Server configuration		
IO Server Name	MODBUS	
Topic Name	A or B or C	
Item Name	ValueName,SlaveAddress	The PLC Address is defined Tag by Tag on serial link (RTU Master)
	ValueName,SlaveAddress,IPAddress	The PLC Address is defined Tag by Tag on TCP link
	ValueName	The Topic PLC address is used

5.3.3.1 ValueName

The Modbus IO Server Tags can be classified following ranges of values. Two types of ranges are used. The two following tables describe the different ranges of value, for each of the two standards.

• **First standard:**

Modbus Type	IO Type	Access	Register address
Coil	Digital Output	R/W	1 → 9999
Contact	Digital Input	R	10001 → 19999
Input Register	Analog Input	R	30001 → 39999
Holding Register	Analog Output	R/W	40001 → 49999
Output Coil*	Digital Output	W	50001 → 59999
Output Registers*	Analog Output	W	60001 → 69999

Table 60: Modbus IO server Tag name convention: first standard

• **Second standard:**

Modbus Type	IO Type	Access	Register address
Coil	Digital Output	R/W	+1 → +65535
Contact	Digital Input	R	+100001 → +165535
Input Register	Analog Input	R	+300001 → +365535
Holding Register	Analog Output	R/W	+400001 → +465535
Output Coil*	Digital Output	W	+500001 → +565535
Output Registers*	Analog Output	W	+600001 → +665535

Table 61: Modbus IO server Tag name convention: second standard

The second standard allows more than 9999 values in each range. Notice the "+" sign before the register value.

* The two last ranges are used with non-standard equipments that do not allow the reading of some of their values. In this case, specifying the address in the "write only" ranges informs the eWON not to read the values after setting them, which is normally done in the other cases. If those registers are read, the returned value will always be 0.

After the numerical value, the characters F, L, I, D or W can be used to specify how to read the value. The following table describes the different character meaning.

Character	Description
W	Reads 1 register considered as 16 bits unsigned integer (DEFAULT IF NOT SPECIFIED)
I	Reads 1 register considered as 16 bits signed integer
D	Reads 2 regs R1, R2 as a DWORD R1 is Less significant, R2 is most significant (32 bits, unsigned) (*)
E	Reads 2 regs R1, R2 as a DWORD R2 is Less significant, R1 is most significant (32 bits, unsigned) (*)
L	Reads 2 regs R1, R2 as a LONG R1 is Less significant, R2 is most significant (32 bits, signed) (*)
M	Reads 2 regs R1, R2 as a LONG R2 is Less significant, R1 is most significant (32 bits, signed) (*)
F	Reads 2 regs R1, R2 as a FLOAT R1 is Less significant, R2 is most significant (32 bits, signed)
H	Reads 2 regs R1, R2 as a FLOAT R2 is Less significant, R1 is most significant (32 bits, signed)

Table 62: the characters used to specify how to read the value

(*) Important: See “Tags are stored as Float” on page 67

When reading a 32 bits value, two consecutive registers or coils are read and combined e.g. 40001L,11 to access in Long representation the reg 1 on the slave 11.

Reading high 32 bits values involve a loss of precision in the mantissa because internally all the values are considered as float by the eWON.

examples :

address	meaning
40001,10	access the <i>Holding Register</i> at address 1 on the UnitID 10
1,11	access the <i>Coil</i> at address 1 on the UnitID 11
+320234,12	access the <i>Input Register</i> at address 20234 on the UnitID 12
40001,100,10.0.0.53	access the <i>Hoding Register</i> at address 1 on the UnitID 100 at IP address 10.0.0.53
40010L,12	access the LONG <i>Holding Register</i> at address 10 (and 11) on the UnitID 12
40008F,15	access the FLOAT <i>Holding Register</i> at address 8 (and 9) on the UnitID 15

Table 63: Modbus address examples

STATUS TAG:

The STATUS Tag is a special Tag that returns information about the current state of the communication for a given device. As for other Tags, the status Tag ItemName is composed of:

Status,Address

If the address is omitted, the Topic address will be used e.g. status,11 points to the status of the slave 11

You can define a status Tag for each address used.

If you use the status MODBUS address, the Tag must be configured as analog:

Tag value	Meaning
0	Communication not initialized. Status UNKNOWN. If no Tag is polled on that device address, the communication status is unknown.
1	Communication OK.
2	Communication NOT OK.

Table 64: MODBUS status values

5.3.3.2 Slave Address

This is the address of the slave device that you want to access. It is a number from 0 to 255.

Example:

30001,11	Polls a RTU device at address 11.
-----------------	-----------------------------------

5.3.3.3 IP Address

This is the IP address of the device on an Ethernet network. It is composed of 4 numbers separated by a dot.

Example:

30001,11,10.0.0.50	Polls a device configured with IP address 10.0.0.50 and with Modbus slave address 11.
---------------------------	---

5.3.3.4 Device specific information

Warning for new users of WAGO modules:

Keep in mind that coil read and write don't use the same address (offset of 0x200); please consult the Wago™ documentation.

Example:

If you use Wago™ systems with two digital inputs and two digital outputs, inputs have addresses 1 and 2, and outputs have the same. The only way to distinguish them it's the read-only access or R/W access.

Tags: station 11

Tag	Modbus address	Comment
MB_DigIn1	10001,11	Digital input module 1 -- read-only
MB_DigIn2	10002,11	Digital input module 2 -- read-only
MB_DigOut1	00001,11	Digital output module 1 for writing - Encode all leading zeroes!
MB_DigOut1Read	10513,11	Digital output module 1 for reading only
MB_DigOut2	00002,11	Digital output module 2 for writing Encode all leading zeroes!
MB_DigOut2Read	10514,11	Digital output module 2 for reading only

Table 65: Wago™ modules - addresses examples

In *View I/O page*, you can change the value of MB_DigOut1 with the update link (set to 1), and if you do that, you view that the value read is always 0.

Why?

Because the eWON reads the value at the WAGO address 1 (thus, DigIn1)! If you want to read the state of the DigOut1, you must read it at WAGO address 513!

The same remark is applied for analog Modbus registers. It's the documented behavior of Wago™-Modbus modules; keep it in mind.