USER MANUAL

MULTIPROTOCOL “KEY-C” GATEWAYS SERIES

MODBUS TO CLOUD (MQTT/HTTP) GATEWAYS

SENECA S.r.l.

Via Austria 26 – 35127 – Z.I. - PADOVA (PD) - ITALY

Tel. +39.049.8705355 – 8705355 Fax +39 049.8706287+

www.seneca.it

ORIGINAL INSTRUCTIONS

CAUTION

SENECA does not guarantee that all specifications and/or aspects of the product and firmware, included in them, will meet the requirements of the actual final application even if the product referred to in this documentation is in compliance with the technological state of the art.
The user assumes full responsibility and/or risk with regard to the configuration of the product to achieve the intended results in relation to the specific installation and/or end application.
SENECA may, with prior agreement, provide consultancy services for the successful completion of the final application, but under no circumstances can it be held responsible for its proper functioning.
The SENECA product is an advanced product, the operation of which is specified in the technical documentation supplied with the product itself and/or can be downloaded, if desired prior to purchase, from the www.seneca.it website.
SENECA has a policy of continuous development and accordingly reserves the right to make and/or introduce - without prior notice - changes and/or improvements to any product described in this documentation.
The product described in this documentation may solely and exclusively be used by personnel qualified for the specific activity and in accordance with the relevant technical documentation, with particular attention being paid to the safety instructions.
Qualified personnel means personnel who, on the basis of their training, competence and experience, are able to identify risks and avoid potential hazards that could occur during the use of this product.
SENECA products may only be used for the applications and in the manner described in the technical documentation relating to the products themselves.
To ensure proper operation and prevent the occurrence of malfunctions, the transport, storage, installation, assembly, maintenance of SENECA products must comply with the safety instructions and environmental conditions specified in this documentation.
SENECA's liability in relation to its products is governed by the general conditions of sale, which can be downloaded from www.seneca.it.
Neither SENECA nor its employees, within the limits of applicable law, will in any case be liable for any lost profits and/or sales, loss of data and/or information, higher costs incurred for goods and/or replacement services, damage to property and/or persons, interruption of activities and/or provision of services, any direct, indirect, incidental, pecuniary and non-pecuniary, consequential damages in any way caused and/or caused, due to negligence, carelessness, incompetence and/or other liabilities arising from the installation, use and/or inability to use the product.

CONTACT US	
Technical support	supporto@seneca.it
Product information	commerciale@seneca.it

Document revisions

DATE	REVISION	NOTES	AUTHOR
14/02/2025	0	First revision	MM
24/02/2025	1	Added chapter on the meaning of LEDs	MM
01/07/2025	2	Added logic rules Added datalogger Added Cloud Seneca Cloudbox 2 Document revision	MM

TABLE OF CONTENTS

1. DESCRIPTION

The Z-KEY-C, R-KEY-LT-C, Z-KEY-2ETH-C products allow you to acquire data from serial or ethernet buses based on Modbus protocols and send them to clouds with the MQTT(s) or http(s) protocols.
Writing from cloud to Modus is also supported.

1.1. MODBUS, MQTT, HTTP PROTOCOLS

The supported Modbus protocols are:
Modbus RTU Master
Modbus RTU Slave
Modbus ASCII Master
Modbus ASCII Slave
Modbus TCP-IP Server
Modbus TCP-IP Client
For further information on these protocols, see the Modbus specification website:
http://www.modbus.org/specs.php

The MQTT protocol supported is version 3.1.1

The HTTP protocol for tags publication on cloud is based on API Rest

The TLS protocol supported is version 1.2

Keys certifications according to X.509 standard

1.2. FEATURES OF THE "KEY" SERIES COMMUNICATION PORTS

PRODUCT	ETHERNET PORTS No.	SERIAL PORTS NO.	ISOLATED SERIAL PORTS
Z-KEY-C	1	2	Yes, both ports
R-KEY-LT-C	1	1	NO
Z-KEY-2ETH-C	2	2	Yes, both ports

2. DEVICE HARDWARE REVISION

With a view to continuous improvement, Seneca updates and makes the hardware of its devices increasingly more sophisticated. It is possible to know the hardware revision of a product via the label on the side of the device.

An example of an R-KEY-LT product label is the following:

The label also shows the firmware revision present in the device (in this case 2.0.1.0) at the time of sale, the hardware revision (in this case) is E00.

To improve performance or extend functionality, Seneca recommends updating the firmware to the latest available version (see the section dedicated to the product on www.seneca.it).

3. FLEX TECHNOLOGY FOR PROTOCOL CHANGE

Starting from the hardware revision indicated in the following table, the KEY series devices include Flex technology.

GATEWAY	***FLEX TECHNOLOGY SUPPORTED BY HARDWARE REVISION***
Z-KEY	"G00"
R-KEY-LT	"E00"
Z-KEY-2ETH	"C00"

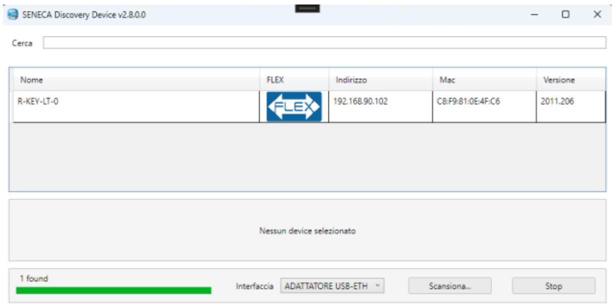
Flex allows you to change the combination of industrial communication protocols supported by the gateways at will from a list of available ones, the development is continuously updated, for a complete list refer to the page: https://www.seneca.it/flex/

Some examples of supported protocols are:

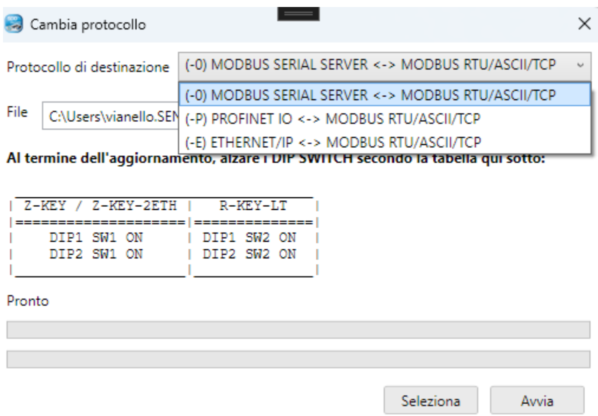
The gateway thus becomes 'universal' and compatible with Siemens, Rockwell, Schneider, etc. systems without the need to purchase different hardware.

3.1. CHANGING PROTOCOLS WITH THE SENECA DISCOVERY DEVICE SOFTWARE

From revision 2.8 the Seneca Discovery Device software identifies the devices that support the "Flex" technology:



For example, in the case in the figure it is possible to press the "Change Protocol" button and select the destination protocol from those in the list:



At the end of the operation, bring (only at the first power-on) the dip switches 1 and 2 to "ON" to force the device to default (see also the chapter "RESETTING THE DEVICE TO ITS FACTORY CONFIGURATION").

Always refer to the user manual of the communication protocol installed in the device by downloading it from the Seneca website.

4. LED MEANING

The devices are equipped with LEDs whose meaning is as follows:

4.1. Z-KEY-C LED

LED	STATUS
PWR	***Steady on***: device powered and IP address set ***Flashing***: IP address not yet set ***Off***: device not powered
COM	***Steady on***: No cloud connection error ***Flashing:*** Cloud connection error (for more details on the error, refer to the webserver status page) ***Off***: device not powered
TX1	***Flashing:*** data transmission on serial port #1 ***Off:*** no transmission on serial port #1
RX1	***Flashing:*** data reception on serial port #1 ***Steady on:*** check wiring on serial port #1 ***Off:*** no reception on serial port #1
TX2	***Flashing:*** data transmission on serial port #2 ***Off:*** no transmission on serial port #2
RX2	***Flashing:*** data reception on serial port #2 ***Steady on:*** check wiring on serial port #2 ***Off:*** no reception on serial port #2
ETH ACT (GREEN)	***Flashing:*** presence of data on ethernet port ***Steady on:*** ethernet port connected but no data present ***Off:*** check wiring of the ethernet port

ETH LNK (YELLOW)	***Steady on:***ethernet cable connected ***Off:***check the wiring of the ethernet port

4.2. R-KEY-LT-C LED

LED	***STATUS***
PWR	***Steady on***: device powered and IP address set ***Flashing***: IP address not yet set ***Off***: device not powered
COM	***Steady on***: No cloud connection error ***Flashing:*** Cloud connection error (for more details on the error, refer to the webserver status page) ***Off***: device not powered
TX	***Flashing:*** data transmission on serial port ***Off:*** no transmission on serial port
RX	***Flashing:*** data reception on serial port ***Steady on:*** check wiring on serial port ***Off:*** no reception on serial port
ETH ACT (GREEN)	***Flashing:*** presence of data on ethernet port ***Steady on:*** ethernet port connected but no data present ***Off:*** check wiring of the ethernet port
ETH LNK (YELLOW)	***Steady on:***ethernet cable connected ***Off:*** check the wiring of the ethernet port

4.3. Z-KEY-2ETH-C LED

LED	***STATUS***
PWR	***Steady on***: device powered and IP address set ***Flashing***: IP address not yet set ***Off***: device not powered
COM	***Steady on***: No cloud connection error ***Flashing:*** Cloud connection error (for more details on the error, refer to the webserver status page) ***Off***: device not powered
TX1	***Flashing:*** data transmission on serial port #1 ***Off:*** no transmission on serial port #1
RX1	***Flashing:*** data reception on serial port #1 ***Steady on:*** check wiring on serial port #1 ***Off:*** no reception on serial port #1
TX2	***Flashing:*** data transmission on serial port #2 ***Off:*** no transmission on serial port #2
RX2	***Flashing:*** data reception on serial port #2 ***Steady on:*** check wiring on serial port #2 ***Off:*** no reception on serial port #2
ET1	***Flashing:*** presence of data on ethernet port #1 ***Steady on:*** ethernet port #1 connected but no data present ***Off:*** check wiring of ethernet port #1
ET2	***Flashing:*** presence of data on ethernet port #2 ***Steady on:*** ethernet port #2 connected but no data present ***Off:*** check wiring of ethernet port #2

5. ETHERNET PORT

The factory configuration of the Ethernet port is:

STATIC IP: 192.168.90.101
SUBNET MASK: 255.255.255.0
GATEWAY: 192.168.90.1

Multiple devices must not be inserted on the same network with the same static IP.

⚠ ATTENTION!
DO NOT CONNECT 2 OR MORE FACTORY-CONFIGURED DEVICES ON THE SAME NETWORK, OR THE DEVICE WILL NOT WORK
(CONFLICT OF IP ADDRESSES 192.168.90.101)

6. FIRMWARE UPDATE

In order to improve, add or optimize the functions of the product, Seneca releases firmware updates on the device section on the www.seneca.it website

The firmware update is performed using Seneca tools or the webserver.

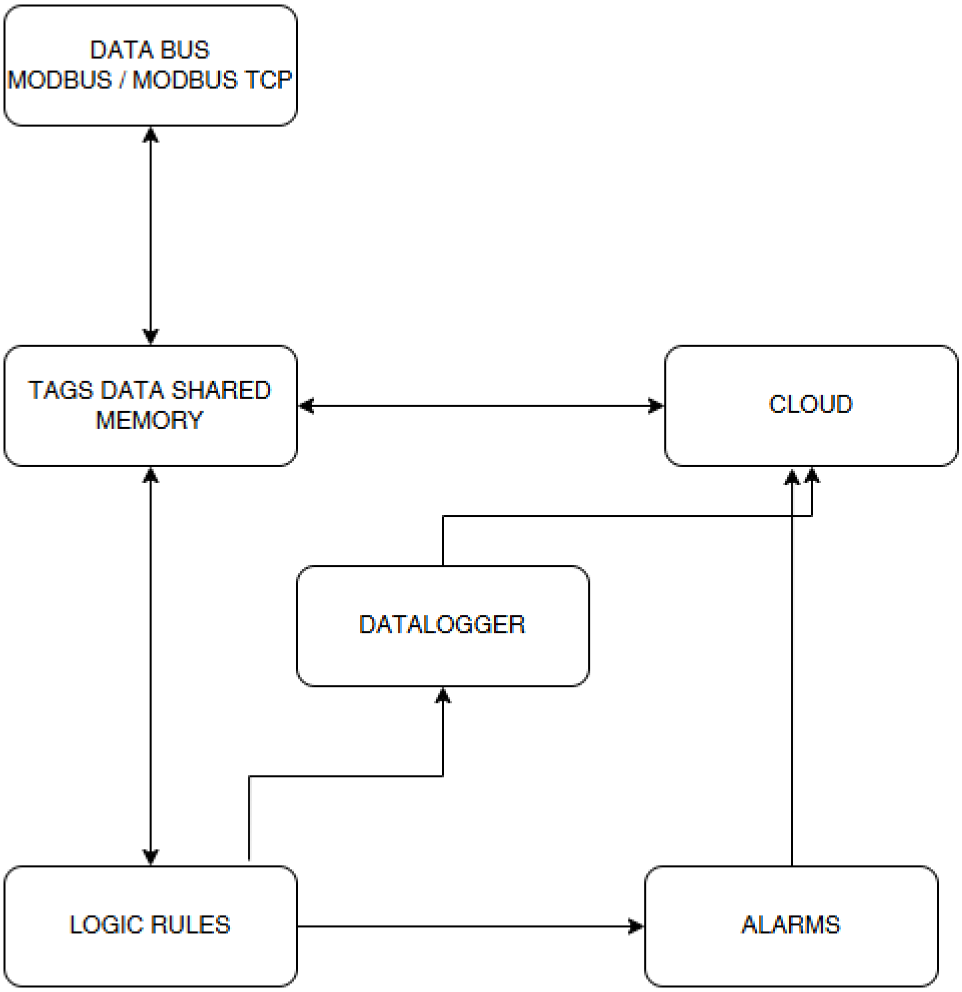
⚠ ATTENTION!
NOT TO DAMAGE THE DEVICE DO NOT REMOVE THE POWER SUPPLY DURING THE FIRMWARE UPDATE OPERATION.

⚠ ATTENTION!
THE FIRMWARE UPDATE WILL DELETE THE DATA ACQUIRED BY THE DATALOGGER. SAVE THE DATA BEFORE PROCEEDING WITH THE UPDATE.

7. DATA ACQUISITION AND PROCESSING, DATA TRANSMISSION AND ALARMS

The devices in the ‘KEY-C’ series allow data to be acquired from buses (via the Modbus RTU/ASCII and Modbus TCP-IP industrial communication protocols). This data is saved in a shared memory and can be processed using scaling or logic rules. Once processed, the data is saved in the internal memory and can be extracted via the web server or sent to the cloud or FTP/Email servers, etc.
Alarms are generated by logic rules and can also be sent to the cloud.

Please refer to the following block diagram:



Data acquisition (Tags) in buses (Data Bus) takes place via Modbus industrial protocols.
This data flows into shared memory (Tags Data Shared Memory), where logic rules perform data processing (Logic Rules).
The data logger acquires the processed data, stores it and sends it to the cloud.
Logic rules generate alarms that can be sent to the Cloud.
The Cloud can access and then write the data already processed in the shared memory (Shared Memory).

Below we will analyse the main components of the block diagram.

7.1. DATA BUS AND MODBUS INDUSTRIAL PROTOCOLS

The data resides in external devices and must be connected via industrial protocols.
The device includes the Modbus RTU/ASCII master and Modbus TCP-IP client protocols so that it can connect to a wide range of third-party manufacturers. The shared memory is also accessible from the outside via the Modbus protocol.

7.1.1. MODBUS PROTOCOLS

Modbus was born as a serial communication protocol by Modicon (a company now part of the Schneider Electric group) to connect their programmable logic controllers (PLCs). It has become a de facto standard in industrial communication, and is currently one of the most widespread connection protocols in the world among industrial electronic devices. In addition to the serial version, Seneca devices also support the Ethernet-based version.
The supported Modbus protocols are:

Modbus RTU Master protocol
Modbus RTU Slave protocol
Modbus TCP-IP Client protocol
Modbus TCP-IP Server protocol

For further information, see website:

https://modbus.org/

Thanks to these protocols it is possible to acquire variables in the memory directly from external Modbus RTU slave or Modbus TCP-IP server devices.

7.2. TAGS DATA SHARED MEMORY

The data acquired by the buses is sent to the shared memory, which can be accessed from outside the device using Modbus protocols.
Each piece of data is identified by a mnemonic name and a type (integer, floating point etc.), thus characterized it takes the name of "Tag".
On these Tags it is possible to perform various types of processing as we will see later in the manual.

7.3. DATA LOGGER AND CACHE

Seneca's 'KEY-C' series gateways include a powerful data logger that allows you to manage up to 300 variables simultaneously (TAG). It is also possible to scale each variable and perform further processing with logic rules. The data acquired by the data logger can then be sent to various clouds or saved in the internal memory.

The logs are stored in flash memory, so if data transfer to the cloud temporarily fails, it can be transferred successfully at a later time.

The data logger's cache fills up after acquiring approximately 23,700 tags. When the limit is reached, the cache 'rotates', i.e. the oldest data is overwritten by the new data.

The devices acquire the set tags and save them in a flash memory. This data can be exported from the web server in CSV format.
The data logger allows the acquisition of up to a maximum of 237,000 samples (equivalent to, for example, 790 acquisitions of 300 tags). Once 237,000 samples have been acquired, the oldest ones are gradually overwritten (circular log).

The maximum duration of the acquisition buffer depends on the acquisition time and the number of tags according to the following formula:

$$Durata\ Buffer[ore] = \frac{237000\ \times\ T_{acq}[s]}{NrTag \times 3600}$$

This gives the following example table:

NO. OF TAGS	***ACQUISITION TIME [s]***	***BUFFER DURATION [h]***
1	30	1975 hrs
10	30	197.5 hrs
100	30	19.75 hrs
1	60	3950 hrs
10	60	395 hrs
100	60	39.5 hrs

7.4. TAG PROCESSING: LOGICAL RULES

KEY-C series devices allow you to define a set of rules that will execute a program. No knowledge of programming languages is required, as the rules are straightforward: ‘If this event occurs, then perform this operation; otherwise, perform that operation.’
For more information, refer to the respective chapters of this manual.

7.5. CONNECTION TO CLOUDS VIA “EASY CLOUD” TECHNOLOGY

The “Easy Cloud” technology is based on the MQTT protocol and allows bidirectional connection with the main available clouds.

7.6. ALARMS

Logic rules are used for TAG alarms, and it is possible to associate a text linked to the alarm.

8. *OPERATING MODE*

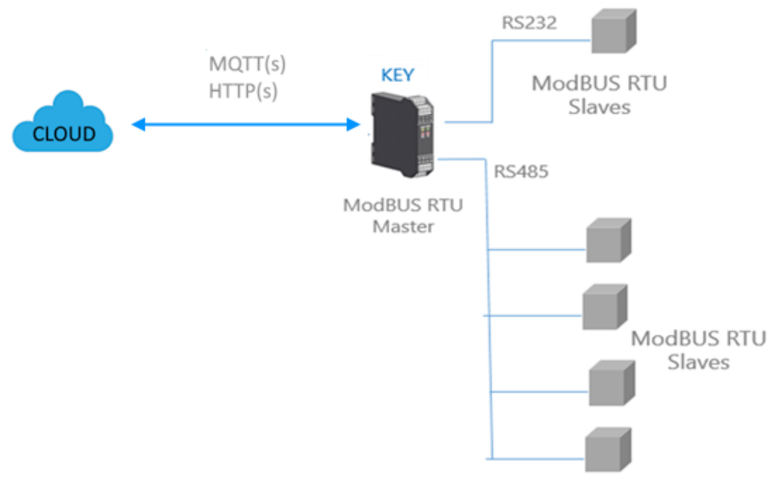
The Gateway operates in the following mode:

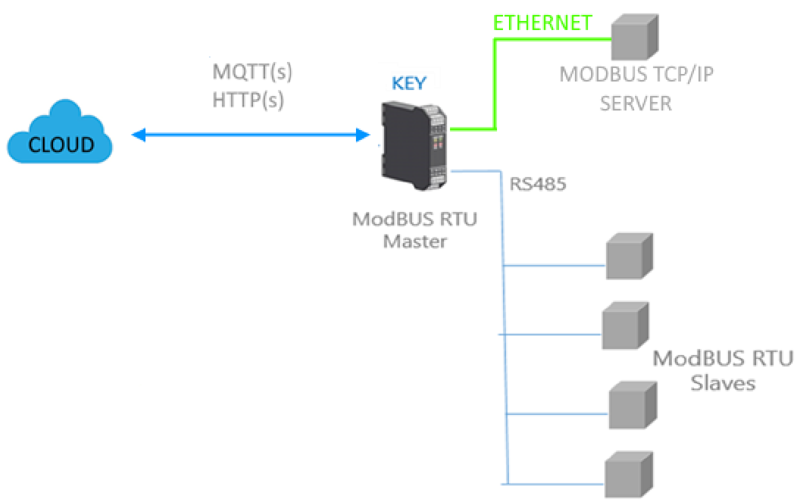
MODBUS SERIAL-ETHERNET MASTER/CLIENT TO CLOUD

8.1. MODBUS MASTER / CLIENT TO CLOUD

You can send data from Modbus RTU/ASCII Slave and/or TCP Server remote I/O to a cloud (and vice versa).

Below are some examples of possible connections:





The Gateway, on the field side, works as a Modbus master / Modbus Client device and on the other side as a client to the MQTT broker or HTTP server via Ethernet.
Modbus requests (read or write commands) are configured in the gateway device.

In addition to serial devices, it is also possible to connect up to 3 remote Modbus TCP-IP servers.
It is also possible to write TAGs (and therefore Modbus registers) from the cloud.
The Gateway always activates a Modbus TCP-IP server at the same time in order to access the shared tag memory.

8.2. SIMPLIFIED TAG DIAGNOSTICS

Tag diagnostics is only available in Modbus TAGs Gateway mode.
Tag diagnostics can also be viewed via the Modbus serial and Ethernet ports: via special Modbus registers.
The first Modbus address, from which the simplified diagnostics starts, is by default 49001 (Holding Register 9000).
Each bit represents a tag with the following meaning:
1 = TAG OK
0 = TAG FAIL
The least significant bit is the status of tag no. 1
The next is the status of tag no. 2 and so on ...
For example the reading of the following registers:
49001 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1
49002 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
Means: TAG 1, TAG 4, TAG17, TAG 18 , TAG 19, TAG 20 OK, all the others in FAIL.
At the start, all tags are in a fail state (all 0).

8.3. EXTENDED TAG DIAGNOSTICS

Tag diagnostics is only available in Modbus TAGs Gateway mode.
When a tag is in an error state it is possible to get more information using extended diagnostics.
Extended diagnostics reserves 1 byte for each tag (since the limit is 500 tags, there are 500 bytes = 250 Modbus registers for extended diagnostics).
This diagnostics is found at the end of the simplified diagnostics (default starting Modbus address is 49033, Holding register 32).

Each Modbus register contains 2 tags, so for example:
49033 TAG02_TAG01
49034 TAG04_TAG03
…
49282 TAG500_TAG499
49283 LAST_LOOP_TIME_COM1 [x1 ms]

49284 LAST_LOOP_TIME_COM2 [x1 ms]

The meaning of the advanced diagnostics byte is:

BYTE VALUE	***MEANING***	***NOTE***
0	OK	The tag is read/written correctly
1	TIMEOUT	The response of the tag timed out, but will be queried again
2	DELAYED	Too many fails, tag polling is delayed (tag will be interrogated again after the configured quarantine time)
3	EXCEPTION	Modbus exception response but the tag will be queried again
4	CRC ERROR	CRC Modbus exception response but the tag will be queried again

For example:

49033 0x0000
49034 0x0002

It means that:

TAGs 1 and 2 are OK (0x00 and 0x00)
TAG 03 is in a delayed state (0x02)
TAG 4 is OK (0x00)

LAST_LOOP_TIME_COMx is a register that contains the last interrogation time of all serial tags (in how many of 10 ms) so, for example:

49283 25
49284 42

It means that the serial 1 loop was 250ms, the serial 2 loop was 420ms.

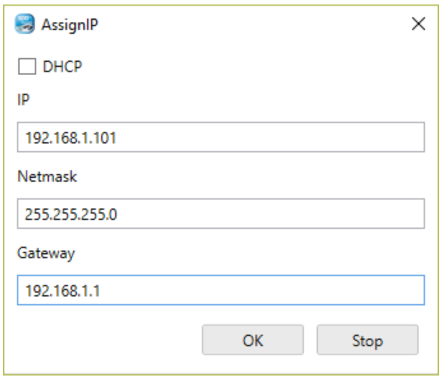
9. “-C” GATEWAY WEBSERVERS

9.1. STEP BY STEP GUIDE FOR THE FIRST ACCESS TO THE WEBSERVER

STEP 1: POWER THE DEVICE AND CONNECT THE ETHERNET PORT

SENECA DISCOVERY DEVICE SOFTWARE STEP 2

If you need to change the IP address of the device (default 192.168.90.101), launch the Seneca Discovery Device software and perform the SCAN, select the device and press the “Assign IP” button, set a configuration compatible with your PC, for example:



Confirm with OK. Now the device can be reached via Ethernet from your PC.

STEP 3 ACCESS TO THE CONFIGURATION WEBSERVER

ENTER your access credentials:
user: admin
password: admin

ATTENTION!
THE WEB BROWSERS WHICH HAVE BEEN TESTED FOR COMPATIBILITY WITH THE DEVICE WEBSERVER ARE:
MOZILLA FIREFOX AND GOOGLE CHROME.
THEREFORE, THE OPERATION WITH OTHER BROWSERS IS NOT GUARANTEED

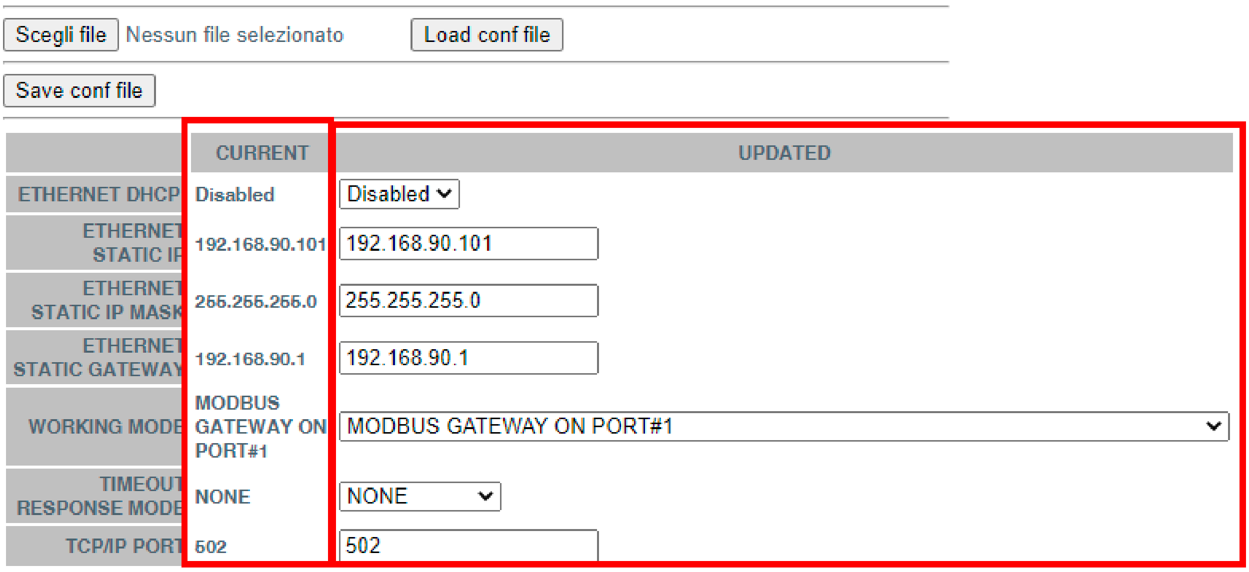
10. WEBSERVER DEVICE CONFIGURATION

ATTENTION!
THE WEB BROWSERS WHICH HAVE BEEN TESTED FOR COMPATIBILITY WITH THE DEVICE WEBSERVER ARE:
MOZILLA FIREFOX AND GOOGLE CHROME.
THEREFORE, THE OPERATION WITH OTHER BROWSERS IS NOT GUARANTEED

ATTENTION!
AFTER THE FIRST ACCESS CHANGE USER NAME AND PASSWORD IN ORDER TO PREVENT ACCESS TO THE DEVICE TO UNAUTHORIZED PEOPLE.

ATTENTION!
IF THE PARAMETERS TO ACCESS THE WEBSERVER HAVE BEEN LOST, TO ACCESS IT, IT IS NECESSARY TO GO THROUGH THE PROCEDURE TO RESET THE FACTORY-SET CONFIGURATION

10.1. "SETUP" PAGE



Scegli file | Nessun file selezionato | Load conf file

Save conf file

	CURRENT	UPDATED
ETHERNET DHCP	Disabled	Disabled
ETHERNET STATIC IP	192.168.90.101	192.168.90.101
ETHERNET STATIC IP MASK	255.255.255.0	255.255.255.0
ETHERNET STATIC GATEWAY	192.168.90.1	192.168.90.1
WORKING MODE	MODBUS GATEWAY ON PORT#1	MODBUS GATEWAY ON PORT#1
TIMEOUT RESPONSE MODE	NONE	NONE
TCP/IP PORT	502	502

The first column represents the name of the parameter, the second column “current” is the current value of the parameter. The last column “updated” is used to modify the current configuration.
When a configuration has been entered it is necessary to confirm it with the "APPLY" button, at this point the new configuration is operational.
If you want to restore the default parameters, click on the "FACTORY DEFAULT" button.

10.1.1. GENERAL CONFIGURATION PARAMETERS

The general configuration parameters are explained below:

DHCP

Disabled: A static Network Configuration is set up
Enabled: The IP address, IP mask and gateway address are obtained from the DHCP server.
The gateway address can be found by the Seneca Discovery Device software.

ETHERNET STATIC IP

Static IP address when the DHCP is disabled

ETHERNET STATIC IP MASK

Mask when the DHCP is disabled

ETHERNET STATIC GATEWAY

Gateway address when the DHCP is disabled

TCP/IP PORT

TCP-IP port for Modbus TCP-IP Server protocol (Up to a maximum of 8 clients can be connected to the gateway)

PORT#n MODBUS PROTOCOL

Select the Modbus RTU or Modbus ASCII serial protocol

PORT#n BAUDRATE

Select the baudrate of the serial port

PORT#n BIT

Select the number of bits for the serial communication.

PORT#n PARITY

Select the type of parity of the serial port (None, Even or Odd)

PORT#n STOP BITS

Set the number of stop bits of the port (1 or 2), note that if parity is set, only 1 stop bit can be used.

PORT#n TIMEOUT [ms]

Set the waiting time for a response from the Modbus slave serial device, after this time without any response there will be a TIMEOUT.

PORT#n DELAY BETWEEN POLLS [ms]

Set the pause between two successive serial Modbus master requests.

PORT#n WRITING RETRIES (Only for Gateway Tags Modbus mode)
Set the number of attempts to write to the TAG(s) before setting the FAIL status.

PORT#n MAX READ NUM (Only for Gateway Tags Modbus mode)
Set the maximum number of registers that can be read with the multiple reading functions (the gateway will optimize readings with this maximum number of registers). It must be adjusted according to the maximum number of registers that can be read at the same time by the slave device.

PORT#1 MAX WRITE NUM (Only for Gateway Tags Modbus mode)
Set the maximum number of registers that can be written with the multiple writing functions (the gateway will optimize writings with this maximum number of registers).

WEB SERVER PORT
Set the TCP-IP port for the Webserver.

WEB SERVER AUTHENTICATION USERNAME
Set the username for accessing the Webserver (if the user name and password are left blank, no authentication is required to access the Webserver)

WEB SERVER AUTHENTICATION PASSWORD
Set the password for accessing the Webserver (if the username and password are left blank, no authentication is required to access the Webserver)

⚠ **ATTENTION!**

CHANGE THE DEFAULT USERNAME AND PASSWORD IN THE WEBSERVER TO RESTRICT ACCESS.

⚠ **ATTENTION!**

IF THE TWO TEXT BOXES IN THE AUTHENTICATION PARAMETERS ARE LEFT BLANK, AUTHENTICATION WILL BE DISABLED.

WEBSERVER HTTPS
It forces the webserver to use the https secure protocol instead of http one

IP CHANGE FROM DISCOVERY
Set whether a user is authorized to change the IP configuration from the "Seneca Discovery Device" software.
DIAGNOSTIC REGISTERS MAPPING
Set the type of register that will contain simplified and advanced diagnostics. It is possible to select between holding registers or input registers.

DIAGNOSTIC REGISTER START ADDRESS

Set the starting address for the diagnostic registers (default offset 9000 -> 49001 in case of holding registers or 39001 in case of input registers)

PORT #n TAGS QUARANTINE [s]

When a TAG is in FAIL it is placed in quarantine and is no longer interrogated for the set time.

MODBUS TCP-IP CLIENT

Enable or not the Modbus TCP-IP clients, the gateway can connect to a maximum of 3 Modbus TCP-IP servers.

TCP-IP PORT SERVER #n (Only if Modbus TCP-IP client is active)

Used to set the TCP-IP server port #n

TCP-IP ADDRESS SERVER #n (Only if Modbus TCP-IP client is active)

Used to set the IP address of the #n server

MODBUS TCP-IP CLIENT TIMEOUT [ms] (Only if Modbus TCP-IP client is active)

Used to set the connection time out for Modbus TCP-IP clients.

MODBUS TCP-IP CLIENT DELAY BETWEEN POLLS [ms] (Only if Modbus TCP-IP client is active)

Set the pause between two successive Modbus TCP-IP client requests.

MODBUS TCP-IP CLIENT WRITING RETRIES (Only if Modbus TCP-IP client is active)

Set the number of attempts to write to the TAG(s) before setting the FAIL status.

MODBUS TCP-IP CLIENT MAX READ NUM (Only if Modbus TCP-IP client is active)

Set the maximum number of registers that can be read with the multiple reading functions (the gateway will optimize readings with this maximum number of registers).

MODBUS TCP-IP CLIENT MAX WRITE NUM (Only if Modbus TCP-IP client is active)

Set the maximum number of registers that can be written with the multiple writing functions (the gateway will optimize writings with this maximum number of registers).

SYNC CLOCK WITH INTERNET TIME

Allows you to enable date/time updating via connection to NTP servers (RFC 5905).

ATTENTION!
EACH TIME THE DEVICE IS SWITCHED OFF, IT MUST BE ABLE TO RECOVER THE DATE/TIME SETTINGS FROM AN NTP SERVER. IF NOT, THE SETTINGS WILL BE RETRIEVED FROM THE LAST LOG ACQUIRED.

NTP SERVER 1 ADDRESS
This is the IP address of the first NTP server (for example 193.204.114.232 for INRIM's NTP)

NTP SERVER 2 ADDRESS
This is the IP address of the second NTP server (in case the first one does not respond)

ATTENTION!
PLEASE NOTE THAT NTP SERVERS USE UDP PORT 123 (WHICH MUST THEREFORE BE OPEN IN THE NETWORK CONFIGURATION USED) ***WATCHDOG ENABLE***

Enable or disable the time restart of gateway.

WATCHDOG TIMEOUT [hours]
Sets the time in hours after which the gateway will reboot (only if the WATCHDOG ENABLE parameter is enabled).

10.1. "DATA LOGGER" PAGE

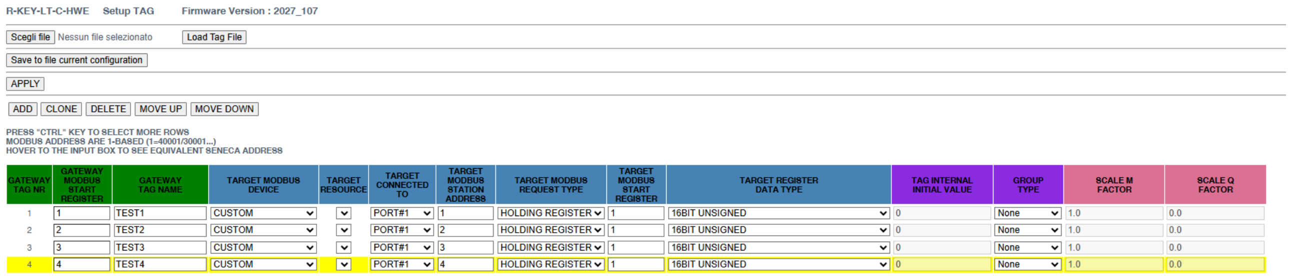
On this page, you can select the data logger acquisition time and download the logs from the device's internal memory in CSV text format.

10.2. "SETUP TAG" PAGE

In Modbus Tags Gateway mode it is necessary to define the Modbus tags (i.e. variables), to do this it is possible to use:

- *The webserver*
- *An excel template*

In this chapter we will explain the configuration of the tag from the webserver.
To edit the TAGs via webserver, access the "Setup tag" section of the navigation menu:



GATEWAY TAG NR	GATEWAY MODBUS START REGISTER	GATEWAY TAG NAME	TARGET MODBUS DEVICE	TARGET RESOURCE	TARGET CONNECTED TO	TARGET MODBUS STATION ADDRESS	TARGET MODBUS REQUEST TYPE	TARGET MODBUS START REGISTER	TARGET REGISTER DATA TYPE	TAG INTERNAL INITIAL VALUE	GROUP TYPE	SCALE M FACTOR	SCALE Q FACTOR
1	1	TEST1	CUSTOM		PORT#1	1	HOLDING REGISTER	1	16BIT UNSIGNED	0	None	1.0	0.0
2	2	TEST2	CUSTOM		PORT#1	2	HOLDING REGISTER	1	16BIT UNSIGNED	0	None	1.0	0.0
3	3	TEST3	CUSTOM		PORT#1	3	HOLDING REGISTER	1	16BIT UNSIGNED	0	None	1.0	0.0
4	4	TEST4	CUSTOM		PORT#1	4	HOLDING REGISTER	1	16BIT UNSIGNED	0	None	1.0	0.0

By selecting a row with the mouse (it will turn yellow, as shown in the figure), you can clone, delete or move it. You can select multiple rows by clicking on the row with the mouse and holding down the CTRL key.

GATEWAY TAG NO.

It indicates the tag number. You can define up to 300 tags.

GATEWAY MODBUS START ADDRESS

Set the address of the Gateway memory location where the TAG is saved, these registers are accessible both from Modbus serial and Modbus TCP-IP.

GATEWAY TAG NAME

Set the mnemonic name of the tag (it will be displayed on the ‘STATUS’ page and will be used in the logic rules).

TARGET MODBUS DEVICE

Select the Modbus RTU slave model from the Seneca device database or select "custom" if you are not using a Seneca Modbus RTU slave.

TARGET RESOURCE

If you are using a Seneca Modbus RTU Slave select the resource name from the Seneca database.

TARGET CONNECTED TO

Select which serial port of the gateway the Modbus RTU slave device is connected to.
(in the case of R-KEY-LT only the COM 1 port is available).
In addition to the serial ports, you can retrieve data from up to 3 remote Modbus TCP-IP servers.
A tag may not be linked to a Modbus acquisition but may be an internal variable that will be used in logical rules.
In this case, select ‘INTERNAL’.

TARGET MODBUS STATION ADDRESS

Defines the Modbus Station Address (also called the Modbus node address) of the slave device.

TARGET MODBUS REQUEST TYPE

Select the type of Modbus register:
Coil
Discrete Input

Holding Register
Input Register

TARGET MODBUS START REGISTER ADDRESS

Defines the starting register of the TAG to be acquired by the Modbus RTU slave.

TARGET REGISTER DATA

Select the TAG variable type:
16 BIT SIGNED: 16 bit variable from -32768 to +32767
16 BIT UNSIGNED: 16 bit variable from 0 to 65535
32 BIT SIGNED MSW: 2 Modbus registers, whose Modbus register with the lower address contains the most significant word, can assume values from -2147483648 to +2147483647
32 BIT UNSIGNED MSW: 2 Modbus registers, whose Modbus register with the lower address contains the most significant word, can assume values from 0 to 4294967295
32 BIT SIGNED LSW: 2 Modbus registers whose Modbus, register with the lower address contains the least significant word, can assume values from -2147483648 to +2147483647
32 BIT UNSIGNED LSW: 2 Modbus registers, whose Modbus register with the lower address contains the least significant word, can assume values from 0 to 4294967295
32 BIT REAL MSW: 2 Modbus registers, whose Modbus register with the lower address contains the most significant word, single precision floating point value (IEEE 758-2008)
32 BIT REAL LSW: 2 Modbus registers, whose Modbus register with the lower address contains the least significant word, single precision floating point value (IEEE 758-2008)
BIT: 1 Boolean Coil or Discrete Input, value true or false.
64 BIT SIGNED MSW 4 Modbus registers whose Modbus register with the lower address contains the most significant word and can assume values from -9223372036854775808 to +9223372036854775807
64 BIT SIGNED LSW 4 Modbus registers whose Modbus register with the lower address contains the least significant word and can assume values from -9223372036854775808 to +9223372036854775807
64 BIT UNSIGNED MSW 4 Modbus registers whose Modbus register with the lower address contains the most significant word and can assume values from 0 to 18446744073709551615
64 BIT UNSIGNED LSW 4 Modbus registers whose Modbus register with the lower address contains the least significant word and can assume values from 0 to 18446744073709551615
16 BIT SIGNED SCALED TO REAL MSW reads a 16-bit signed register from Modbus and converts it to a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".
16 BIT SIGNED SCALED TO REAL LSW reads a 16-bit signed register from Modbus and converts it to a real value where the lower address contains the least significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".
16 BIT UNSIGNED SCALED TO REAL MSW reads a 16-bit unsigned register from Modbus and converts it to a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

16 BIT UNSIGNED SCALED TO REAL LSW reads a 16-bit unsigned register from Modbus and converts it to a real value where the lower address contains the least significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".
32 BIT SIGNED MSW SCALED TO REAL MSW reads two registers, interpreting them as 32 bit signed from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT SIGNED MSW SCALED TO REAL LSW reads two registers, interpreting them as 32 bit signed from Modbus, and converts them into a real value where the lower address contains the least significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT UNSIGNED MSW SCALED TO REAL MSW reads two registers, interpreting them as 32 bit unsigned from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT UNSIGNED MSW SCALED TO REAL LSW reads two registers, interpreting them as 32 bit unsigned where the lower address contains the most significant word from Modbus, and converts them into a real value where the lower address contains the least significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT SIGNED LSW SCALED TO REAL MSW reads two registers, interpreting them as 32 bit signed where the lower address contains the least significant word from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT SIGNED LSW SCALED TO REAL LSW reads two registers, interpreting them as 32 bit signed where the lower address contains the least significant word from Modbus, and converts them into a real value where the lower address contains the least significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT UNSIGNED LSW SCALED TO REAL MSW reads two registers, interpreting them as 32 bit unsigned where the lower address contains the least significant word from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT UNSIGNED LSW SCALED TO REAL LSW reads two registers, interpreting them as 32 bit unsigned where the lower address contains the least significant word from Modbus, and converts them into a real value

where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".
64 BIT SIGNED MSW SCALED TO REAL MSW reads 4 registers, interpreting them as 64 bit signed where the lower address contains the most significant word from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

64 BIT SIGNED MSW SCALED TO REAL MSW reads 4 registers, interpreting them as 64 bit signed where the lower address contains the most significant word from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

64 BIT UNSIGNED MSW SCALED TO REAL MSW reads 4 registers, interpreting them as 64 bit unsigned where the lower address contains the most significant word from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

64 BIT UNSIGNED MSW SCALED TO REAL LSW reads 4 registers, interpreting them as 64 bit unsigned where the lower address contains the most significant word from Modbus, and converts them into a real value where the lower address contains the least significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

64 BIT SIGNED LSW SCALED TO REAL MSW reads 4 registers, interpreting them as 64 bit signed where the lower address contains the least significant word from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

64 BIT SIGNED LSW SCALED TO REAL LSW reads 4 registers, interpreting them as 64 bit signed where the lower address contains the least significant word from Modbus, and converts them into a real value where the lower address contains the least significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

64 BIT UNSIGNED MSW SCALED TO REAL MSW reads 4 registers, interpreting them as 64 bit unsigned where the lower address contains the most significant word from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

64 BIT UNSIGNED LSW SCALED TO REAL LSW reads 4 registers, interpreting them as 64 bit unsigned where the lower address contains the least significant word from Modbus, and converts them into a real value where the lower address contains the least significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT REAL MSW SCALED TO REAL MSW reads 2 registers, interpreting them as real values, where the lower address contains the most significant word from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT REAL LSW SCALED TO REAL MSW reads 2 registers, interpreting them as real values, where the lower address contains the least significant word from Modbus, and converts them into a real value where the lower address contains the most significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

32 BIT REAL LSW SCALED TO REAL LSW reads 2 registers, interpreting them as real values, where the lower address contains the least significant word from Modbus, and converts them into a real value where the lower address contains the least significant word. If desired, you can also scale the tag before conversion according to the formula: TAG SCALED = (TAG * "SCALE M FACTOR")+"SCALE Q FACTOR".

N.B. This field is automatically filled in if a Seneca slave device has been selected in the "TARGET MODBUS DEVICE" field.

ATTENTION!

All 32-bit values are stored in 2 consecutive registers, for example:
The 32-bit unsigned MSW TAG 1 Totalizer is stored in the addresses 40016 and 40017:
The most significant word is 40016, the least significant is 40017.
So the 32bit value is obtained from the following relationship:

$$Totalizer1 = (40017) + (Reg\ (40016) \times 65536)$$

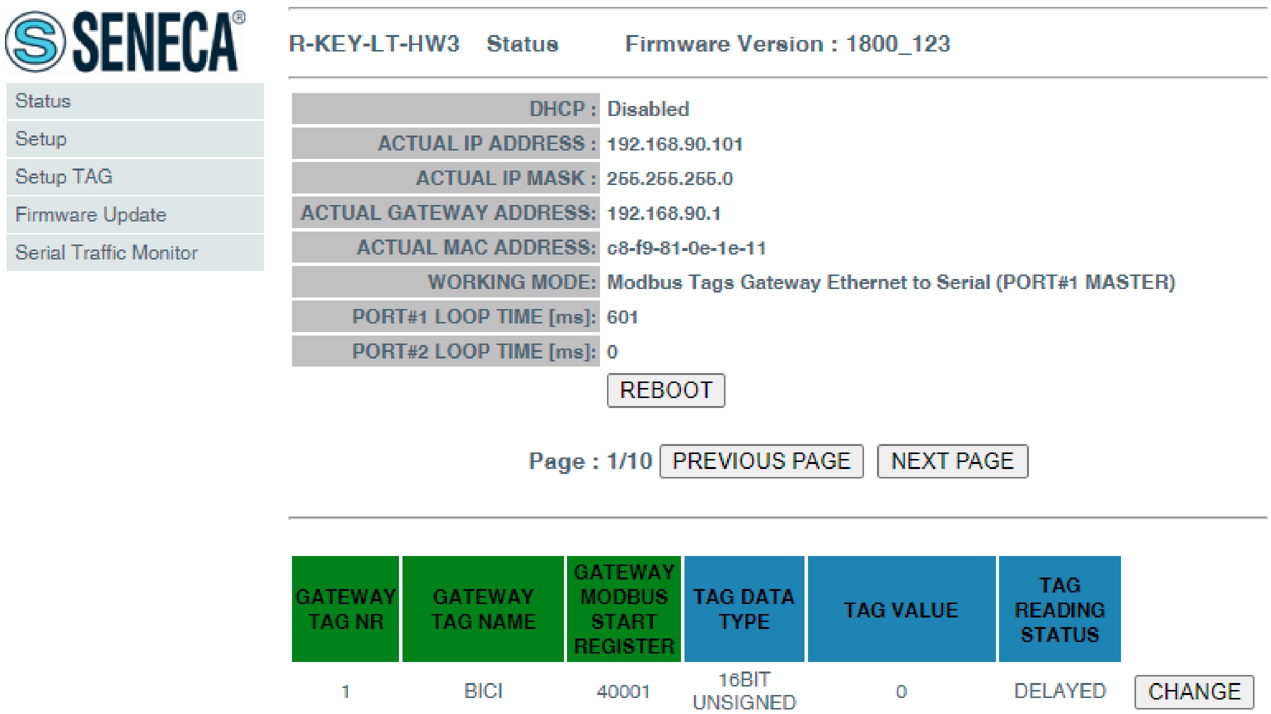
Similarly, all 64-bit values are stored in 4 consecutive registers.

INTERNAL TAG INITIAL VALUE

Select the initial value of the internal Tag (i.e. a tag that does not come from Modbus but is defined as an internal variable for logic rules).

10.2.1. Real-time view of the Modbus Gateway: "STATUS" PAGE

Once the TAGs are configured, it is possible to view the status of the Modbus communication in real time, from the Status section of the navigation menu.
The live view will show the current network configuration, operation mode and TAG information.



Tag information includes: Name, Modbus Gateway address, value, and status.

The following legend applies to the status field:

OK = TAG free of errors
FAIL_TO = TAG reading time out
DELAYED = Once the set retry number has been reached, the polling of the tag is delayed (the tag will be interrogated again after the configured quarantine time)
EXC = Modbus protocol exception response

10.3. "SETUP TEXT MESSAGE" PAGE

Here you can define the text of the alarms that will be sent to the cloud.

The message text can only contain ASCII characters.
It is possible to use the {TAG_NAME} syntax to include the current value of a tag in the text.

For example the message text:

"WATER LEVEL ={LEVEL} m"

Will provide a text with the tag value as text, if the tag "LEVEL" is 1,232 you will have:

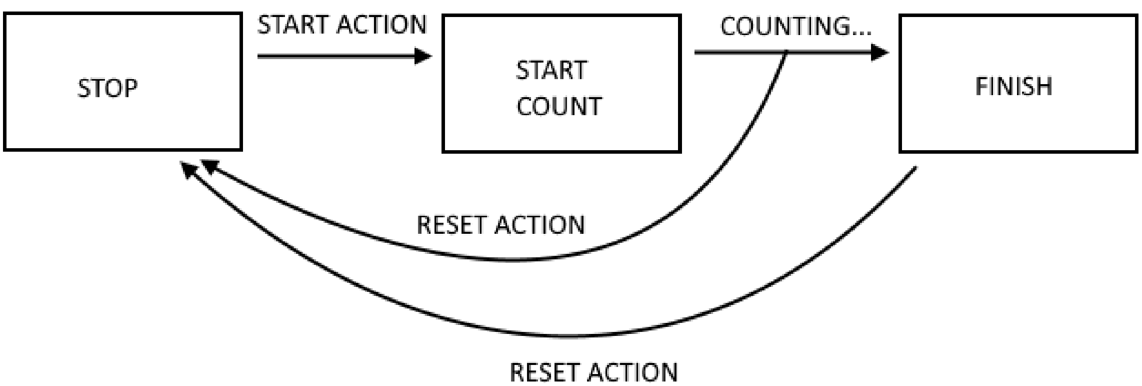
WATER LEVEL = 1.232 m

This syntax can be used more than once in a message text.
Each message has an ID field which is used to associate the message with the alarm in the logical rules.

10.4. "SETUP TIMER" PAGE

This section allows you to define the timers to be used in logic rules.
The ID represents the mnemonic of the timer that must be used in the rules.
"Enable" selects whether the timer is active or not.
"Duration" is the activation value in [ms].

Note
The timers are in stop mode by default, they need an action to start and an action to restore, according to the following scheme:



10.5. "SETUP RULES" PAGE

In this section you can define a set of logical rules that will implement a program.

Please note that Tag writing takes place 'During execution', i.e. tag writing is performed immediately after the write action has been executed.

To configure a rule, the following parameters are available:

10.5.1. RULE CONFIGURATION

ID
Rule execution order (1 = First rule to be executed)

Enabled
Indicates whether the rule is enabled or should be excluded from execution

Description
Mnemonic textual description of the rule

Period [ms]
If the value is = 0, actions are executed only if there is a change in the result of the "OR / AND" (i.e. on change of state).

If the value is different from 0 ms the actions are performed trying to respect the inserted timing.

ATTENTION!

Use appropriate period values for sending actions via MQTT/HTTP.

NOTE:
If Period is > 0 the actions are always performed in "repeat" mode

10.5.2. IF CONDITION: TYPE

This section defines the type of condition, the following types are possible:

None
No conditions to be assessed

Always True
The If condition is always true.
Please note that the rule is only executed once if Period is = 0 ms or if the ‘ACTION MODE’ field of the action is set to ‘One Time’ mode.
If you need to execute a rule at each cycle, you need to put the actions in "repeat mode".
If you need to run a rule over time (every x ms), you must set Period > 0ms.

Always False
The If condition is always false.
This can be used for debugging purposes (by quickly blocking the condition).

Digital Tag
The condition depends on the state of a digital tag:

Field	***Meaning***
Tag	Selects the tag to be used for the condition
Operator	Only "=" may apply
Tag / Constant value	Selects whether the comparison is between another digital tag or a constant boolean value (TRUE or FALSE)

Analog Tag
The condition depends on a comparison with an analog TAG

Field	***Meaning***
Tag	Selects the tag to be used for the condition
Operator	It may be: “=” “>” “<” “>=”

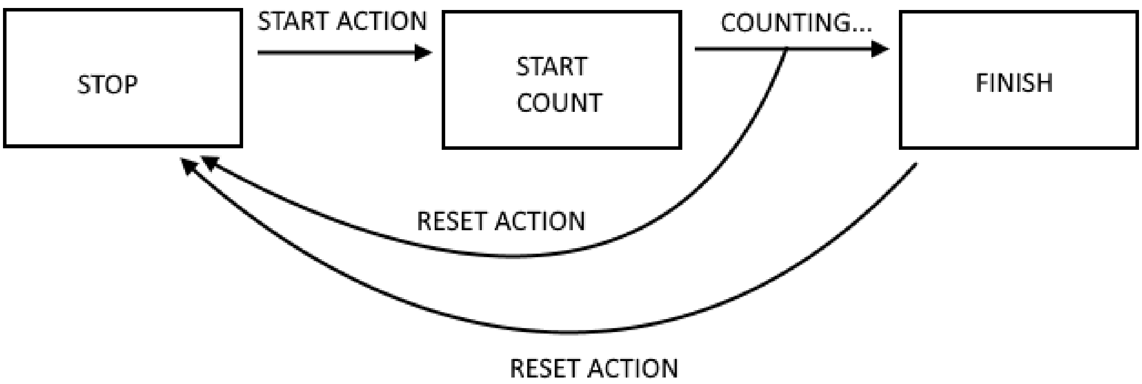
	“<=”
Tag / Constant value	Selects whether the comparison is between another analog tag or a constant value

Timer

The condition depends on the state of the selected timer

Field	***Meaning***
ID	Selects the timer ID to use
Expired	It can be: "OFF" or "ON" With "ON" the condition is only true when the timer expires (FINISH status). With "OFF" the condition is true until the timer is in STOP or COUNTING. When the timer is in FINISH state the condition becomes false.

The operation of the Timer is shown in the following diagram:



Scheduler

The condition depends on the set scheduler (calendar):

Field	***Meaning***
Type	It may be: Every Day, Every week, Every Month Every Day: the condition is true every day at the configured hour and minute

	Every Week: the condition is true once a week on the selected day of the week at the selected hour and minute Every Month: the condition is true once a month on the selected day of the month at the selected hour and minute
Day	If the type is Weekly sets the day of the week: 0 = Sunday 1 = Monday 2 = Tuesday 3 = Wednesday 4 = Thursday 5 = Friday 6 = Saturday If the type is Monthly: Selects the day of the month from 1 to 31
Hour	Hours
Minute	Minutes

Rule Status
The condition depends on whether a rule is enabled or not:

Field	***Meaning***
ID	Selects the rule ID
Enabled	Selects between "enabled" or "disabled" If "Enabled" the condition is REAL if the selected rule is enabled. If "Disabled" the condition is REAL if the selected rule is disabled.

Bitmask
The condition depends on masking a tag with a hexadecimal constant:

Field	***Meaning***
Tag	Selects the tag to apply the bitmask to from a list containing all tags with data type "16Bit Unsigned"
Mask	The bit mask represented as a string of 4 hexadecimal digits

<u>The "Bit mask" condition is TRUE if the AND operation bit by bit between the Tag and the Data Mask is different from 0; FALSE otherwise.</u>

Example:
Tag=0x1233 (hexadecimal) = 0b 0001 0010 0011 0011 (binary)

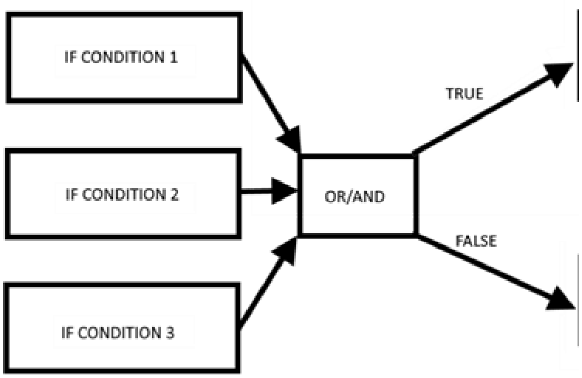
Mask=0x8001 (hexadecimal) = 0b 1000 0000 0000 0001 (binary)
It means that the mask analyses bit0 (least significant) and bit 15 (most significant) of the Tag.
The AND bit by bit provides:

0001 0010 0011 0011
1000 0000 0000 0001

0000 0000 0000 0001
So the condition is TRUE.

10.5.3.IF CONDITION OPERATOR

The "IF conditions" can be combined together in "OR" or "AND" logic, in practice:



The "IF conditions" linked together by "OR" go to the TRUE state if at least one of the conditions is true.
The "IF conditions" linked together by "AND" only go to the TRUE state if all of them are true.

More details are given in the following table:

IF CONDITION 1	*IF CONDITION 2*	*IF CONDITION 3*	*"OR"*	*"AND"*
FALSE	FALSE	FALSE	FALSE	FALSE
FALSE	FALSE	TRUE	TRUE	FALSE
FALSE	TRUE	FALSE	TRUE	FALSE
FALSE	TRUE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	TRUE	FALSE	TRUE	FALSE
TRUE	TRUE	TRUE	TRUE	TRUE

10.5.4.THEN/ELSE ACTION

In this section you can define the action that must be performed if the conditions result in TRUE (THEN action) or FALSE (ELSE action).

NONE

No action to take

Digital Tag

Performs a write to a digital tag.

Field	***Meaning***
Action Mode	Allows you to select between "One Time" or "Repeat". With "One Time" the action is executed only if there is a change in the result of the OR / AND conditions. With "Repeat" the action is executed at every loop (if the rule is enabled and if there is no configured period).
Destination Tag	This is the tag where the calculated TRUE/FALSE result is copied
Operator	This is the Boolean operator to use, selected from =, NOT, OR etc ...
Source Tag 1 / Constant value 1	Selects the first tag to use in the boolean calculation. It is also possible to use a boolean constant
Source Tag 2 / Constant value 2	Select the second Tag if the operator needs 2 inputs (For example operator "OR"). It is also possible to use a boolean constant

Analog Tag

Performs a write to an analog type Tag.

Field	***Meaning***
Action Mode	Select from "One Time" or "Repeat". With "One Time" the action is executed only if there is a change in the result of the OR / AND conditions. With "Repeat" actions are performed at each loop (if the rule is enabled and there is no configured period).
Destination Tag	This is the tag where the calculated result is copied to
Operator	It is the mathematical operator to use, you can select from: "=" copies the source tag 1 or the constant value 1 to the destination tag Example: Destination tag = Origin tag 1 Or Target tag = constant value 1

	"+ =" Add the value of the source tag1 or the constant value 1 to the target tag and copy the result to the target tag. Example: Destination tag = Destination tag + Origin tag 1 "- =" Subtracts the value of the source tag1 from the target tag and copies the result to the target tag. Example: Destination tag = Destination tag - Origin tag 1 "* =" Multiply the target tag by the value of source tag 1 and copy the result to the target tag. Example: Destination tag = Destination tag * Origin tag 1 "/ =" Splits the target tag with the source tag value 1 and copies the result to the target tag. Example: Destination tag = Destination tag / Origin tag 1 "% =" Calculates the rest of the division from the target tag and the value of the source tag1 and copies the result to the target tag. (Note that 53% 7 = 4) Example: Destination tag = Destination tag% Source tag1 "abs" Calculates the absolute value of Source Tag 1 or Constant value 1 and copies the result to the Destination Tag (Note that abs (-4) = 4) Example: Target tag = abs (Source tag 1)

	"Sqrt" Calculates the square root value of source tag 1 or constant value 1 and copies the result to the target tag. (Note that sqrt (9) = $\sqrt{9}$ = 3) Example: Destination tag = sqrt (origin tag 1) "Sqr" Calculates the square value of the source tag 1 or constant value 1 and copies the result to the target tag. (Note that sqr (3) = 3^2 = 9) Example: Destination tag = sqr (origin tag 1) "Log" Calculates the decimal logarithm of source tag 1 or constant value 1 and copies the result to the target tag. (Note that log (3) = 0.4771212) Example: Destination tag = log (origin tag 1) "Ln" Calculates the natural logarithm of the source tag 1 or constant value 1 and copies the result to the target tag. (Note that ln (3) = 1.09861228867) Example: Target tag = ln (Source tag 1) "Exp" Calculate the number of Euler elevated to Source Tag 1 or Constant value 1 and copy the result to the Destination Tag. Please note that: ln (exp 3) = 3 Example: Destination tag = expiration (origin tag 1) "+" Adds Source Tag 1 or Constant value 1 to the value of Source Tag 2 or Constant value 2 and copies the result to the Destination Tag. Example: Target tag = Source tag 1+ Source tag 2

	"-" Subtracts the source tag 1 or constant value 1 with the value of source tag 2 or constant value 2 and copies the result to the target tag. Example: Destination tag = Origin tag 1- Origin tag 2 "*" Multiply the source tag 1 or constant value 1 with the source tag 2 or constant value 2 and copy the result to the target tag. Example: Target tag = Source tag 1 * Source tag 2 "/" Splits the source tag 1 or constant value 1 with the source tag 2 or constant value 2 and copies the result to the target tag. Example: Target Tag = Source Tag 1 / Source Tag 2 "%" Calculates the rest of the division between source tag 1 or constant value 1 and source tag 2 or constant value 2 and copies the result to the target tag. (Note that 53% 7 = 4) Example: Target tag = Source tag 1% Source tag 2 "Pow" Calculates the Source Tag1 or Constant value 1 elevated to the power of the Source Tag2 / Constant value 2 and copies the result to the destination tag. Example: Target tag = (Source Tag1) ^ (Source Tag2)
Source Tag 1 / Constant value 1	Selects the tag to be used as input 1 for the operator used. You can also use a constant value.
Source Tag 2 / Constant value 2	Selects the Tag to use as input 2 in the calculation if the operator needs 2 inputs. A constant value can also be used

Timer

It is possible to select the action to be performed in the selected timer

Field	*Meaning*
Id	Selects the timer from those configured
Action	Selects the type of action to perform on the selected timer. "Start" performs the start action on the selected timer "Reset" performs the reset action on the timer to the stop state

Rule Status

The action enables or disables a rule.

Field	*Meaning*
Id	Selects the rule
Enable	Selects whether or not the action should enable the selected rule: "OFF" disables the selected rule "ON" enables the selected rule

Bitmask

This action allows you to set a configurable number of bits of a given tag to the value 1 or to the value 0.

Field	*Meaning*
Action Mode	Selects from "One Time" or "Repeat". With "One Time" the action is executed only if there is a change in the result of the OR / AND conditions. With "Repeat" the action is executed at every loop (if the rule is enabled and if there is no configured period).
Destination Tag	It is the tag in which the result of the action is copied, the tag must be of type "16 bit unsigned"
Source Tag	Selects the tag to use in the calculation. It is also possible to insert the same source tag and destination tag in order to perform the action on the same TAG. The tag must be of the "16 bit unsigned" type
Mask	It is the mask in hexadecimal format that allows the masking of the bits to be controlled.
Action	You can choose between "Set" or set the bits to 1, or "Reset" or set the bits to 0.

Send Message

Performs a write to a string-type Tag.

Field	*Meaning*
Message	This is the ID of the message to be sent via MQTT/HHTP.

10.6. "CLOUD SETUP" PAGE

On this page, you can configure the connection to the cloud of the configured Tags.

CLOUD PROTOCOL

Selects the protocol to use between MQTT and http

CLOUD PERIODIC SENDING INTERVAL [s]

Selects the tags sending time to the cloud

CLOUD SERVER ADDRESS

Selects the cloud address to be connected to

CLOUD SERVER PORT

Selects the server port

MQTT CLIENT ID/HTTP PATH

Defines the Client ID used in the MQTT protocol *or the publication path on HTTP server*

MQTT WEBSOCKET

Allows you to activate MQTT communication via Websockets

MQTT KEEP ALIVE INTERVAL [s]

This parameter defines Keep alive which ensures that the connection between the broker and client is still open and that the broker and client are aware that they are connected. When the client establishes a connection to the broker, it tells the broker a time interval in seconds. This interval defines the maximum period of time during which the broker and client may not communicate with each other.

MQTT CLEAN SESSION

This parameter defines the "clean session". When the clean session flag is set to true, the client does not want a persistent session. If the client disconnects for any reason, all information and messages queued from a previous session are lost.

MQTT MESSAGE RETAIN

Usually if a publisher publishes a message on a topic to which no one is subscribed, the message is simply discarded by the broker. However, the publisher can tell the broker to keep the last message of that topic.

MQTT QUALITY OF SERVICE [QOS]

This parameter defines the QOS of the MQTT protocol.
Can be selected from
QOS 0 (once only, without ack)
QOS 1 (at least once, with ack)

QOS 2 (once only, with ack and resend)

CLOUD AUTHENTICATION
This parameter defines whether user/password authentication should be used to access the cloud

CLOUD AUTHENTICATION USER
Broker or server username

CLOUD AUTHENTICATION PASSWORD
Broker or server password

CLOUD SSL/TLS
Defines whether to enable the SSL/TLS 1.2 encrypted security protocol

CLOUD CLIENT CERTIFICATE REQUIRED
Defines whether to manage x.509 certificates for the SSL/TLS connection

CLOUD CLIENT CERTIFICATE VALIDITY CHECK
If activated, it verifies the certificates are valid

CLOUD LOG ON CHANGE
Updates values on broker or server only upon change and no longer over time

CLOUD PUBLISH MULTIPLE TAGS
For MQTT protocol, this parameter defines whether the publish contains multiple tags or whether the device should send a publish for each tag.
For HTTP protocol, this parameter defines whether the post contains multiple tags or whether the device should send a post for each tag.

CLOUD PUBLISH TOPIC FOR LOGS
Selects the topic name for the logs using the following table:

%c	Device Client ID
%m	Device MAC Address
%j[field]	Adds double quotes " to [field]. The double quotes represent a string in JSON

For example:

If:
Device Client ID = Padova13
Publish Topic for Logs = seneca/%c/data

The data logs will be sent to the topic: Seneca/Padova13/data

CLOUD PUBLISH PAYLOAD FOR LOGS
Selects the format to be used for the payload using the following table:

%c	Device Client ID
%m	Device MAC Address
%d	date-time
%t	timestamp (number of seconds from 01/01/1970)
%tms	timestamp (number of milliseconds from 01/01/1970)
%b	bulk (format specified in "Publish Bulk Format")
%n	Tag name (only for "Publish Bulk Format")
%i	Unique ID of the variable
%v	Tag value (only in "Publish Bulk Format")
%j[field]	Adds double quotes " to [field]. The double quotes represent a string in JSON

Note: the %i placeholder adds a unique ID to the variable to be published according to the TAG order (see Tag view page)

CLOUD PUBLISH BULK FORMAT

Selects the format for "bulk mode" according to the following table:

%c	Device Client ID
%m	Device MAC Address
%d	Date/Time
%t	timestamp (number of seconds from 01/01/1970)
%n	Tag name (only for "Publish Bulk Format")
%v	Tag value (only in "Publish Bulk Format")
%j[field]	Adds double quotes " to [field]. The double quotes represent a string in JSON

CUSTOM CLOUD

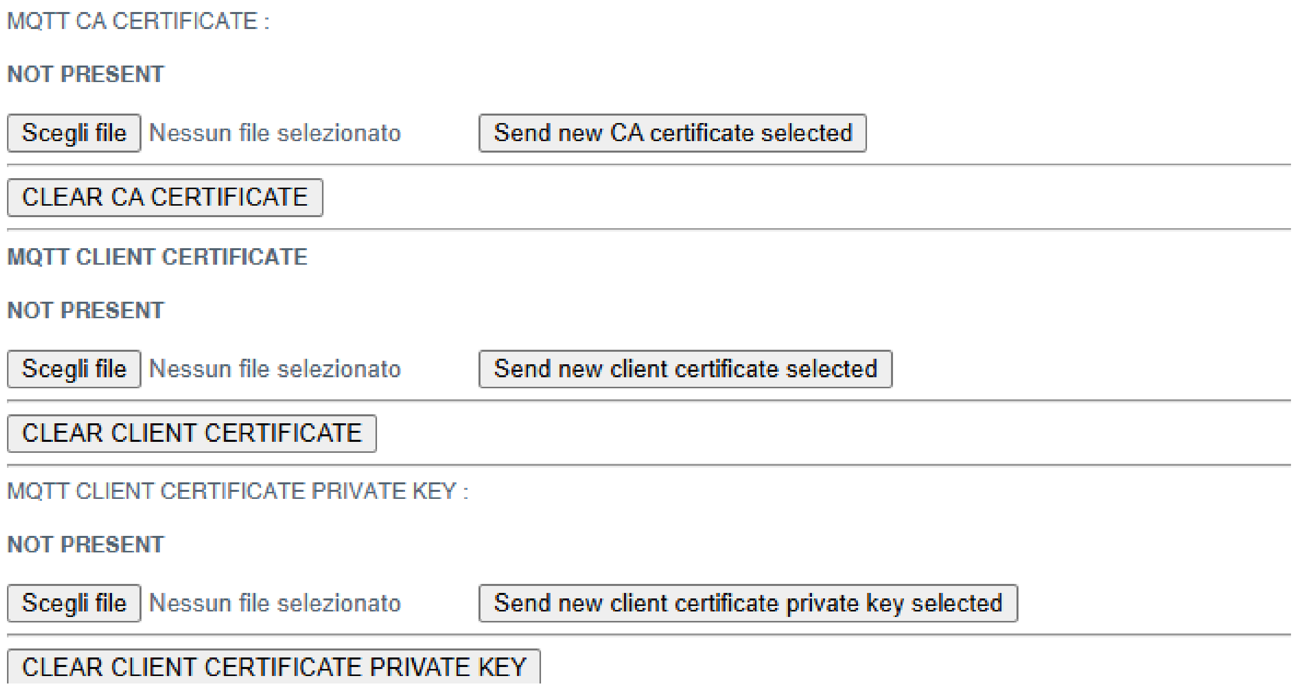
If the MQTT cloud protocol is selected, you can choose between the clouds:

Direl and ONBOARD

Currently, you can configure:
Generic: Through the device's MQTT configurability, it is possible to connect to virtually any cloud
Direl ADM: Sets up the device to connect to the Direl ADM cloud
On-Board: Sets up the device to connect to the On-Board cloud

To add other clouds to the list, you can make a request to Seneca.

At the bottom of the page, you can upload the certificates and key for connecting to the MQTT server in .pem format.



10.6.1.1. EXAMPLE

Let's say we want to send the logs of two tags: tag1 and tag2, with the following configuration:

Client ID = “Test”
Publish Topic for Logs = seneca/%c/data
Publish Payload for Logs = {"type": "data", "message": {"device": %jc, "date": %t, "signals": [%b]}}
Publish Bulk Format = {"name": %jn, "value": %v, "valid" : %i}

You will get: on “Seneca/Test/data” topic the following Payload:
{"type": "data", "message": {"device": “Test”, "date": 1750942723, "signals": [{"name": “tag1”, "value": 1234, "valid" : 1}, {"name": “tag2”, "value": 5678, "valid" : 1}]}}

10.6.2. DIREL ADM4.0

The parameters for the Direl cloud (https://www.direl.it/) are as follows:

Field	Meaning
Enable	Enables or disables the connection to the Direl ADM4.0 cloud
Username for Commands	This is the username for writing access from the cloud to the device
Password for Commands	It is the password for writing access from the cloud to the device

10.6.3. ONBOARD

Onboard is the cloud of innovation system s.r.l., for more information refer to the site:

https://www.onsystem-iot.com/onboard

The parameters for the connection are:

Field	Meaning
Enable	Enables or disables the connection to the Onboard cloud
Username	This is the username for accessing the cloud
Password	This is the password for accessing the cloud

CLOUD SUBSCRIBE TOPIC FOR COMMANDS
To write a tag via MQTT, the device must receive a PUBLISH from the cloud itself with the format indicated in this field.

MQTT CA CERTIFICATE FILE (.pem)
File that represents the Root CA Certificate in .pem format.

MQTT/HTTP SERVER CERTIFICATE FILE (.pem)
File that represents the Client Certificate in .pem format.

MQTT CLIENT PRIVATE KEY FILE (.pem)
File that represents the Client key in .pem format.

10.6.1. SENECA CLOUDBOX 2

Cloudbox 2 is the cloud of Seneca s.r.l. For more information refer to the site:

https://www.seneca.it

The parameters for the connection are:

Field	Meaning
Enable	Enables or disables the connection to the Onboard cloud
Username	This is the username for accessing the cloud
Password	This is the password for accessing the cloud

CLOUD SUBSCRIBE TOPIC FOR COMMANDS
To write a tag via MQTT, the device must receive a PUBLISH from the cloud itself with the format indicated in this field.

MQTT CA CERTIFICATE FILE (.pem)
File that represents the Root CA Certificate in .pem format.

MQTT/HTTP SERVER CERTIFICATE FILE (.pem)
File that represents the Client Certificate in .pem format.

MQTT CLIENT PRIVATE KEY FILE (.pem)
File that represents the Client key in .pem format.

10.7. "FIRMWARE UPDATE" PAGE

Allows you to update the firmware of the device.

⚠ ATTENTION!
NOT TO DAMAGE THE DEVICE DO NOT REMOVE THE POWER SUPPLY DURING THE FIRMWARE UPDATE OPERATION.

⚠ ATTENTION!

THE FIRMWARE UPDATE WILL DELETE THE DATA ACQUIRED BY THE DATALOGGER. SAVE THE DATA BEFORE PROCEEDING WITH THE UPDATE.

10.8. "UTC TIME SETUP" PAGE

It allows you to set time and date.

ATTENTION!

EACH TIME THE DEVICE IS SWITCHED OFF, IT MUST BE ABLE TO RECOVER THE DATE/TIME SETTINGS FROM AN NTP SERVER. IF NOT, THE SETTINGS WILL BE RETRIEVED FROM THE LAST LOG ACQUIRED.

10.9. "CERTIFICATE/DATABASE UPDATE" PAGE

On this page you can upload X.509 certificates for the webserver to the device (if https mode is enabled) and update the Seneca device database.

10.10. "SERIAL TRAFFIC MONITOR" PAGE

The Serial Traffic Monitor page of the webserver shows the serial packets that the gateway is receiving and transmitting for line debugging:

START/STOP TRAFFIC MONITOR ENABLED

116	RECEIVE	01 03 00 00 00 01 84 0a
14	SEND	01 03 02 12 34 b5 33
114	RECEIVE	01 03 00 00 00 01 84 0a
16	SEND	01 03 02 12 34 b5 33
112	RECEIVE	01 03 00 00 00 01 84 0a
18	SEND	01 03 02 12 34 b5 33
109	RECEIVE	01 03 00 00 00 01 84 0a
11	SEND	01 03 02 12 34 b5 33
117	RECEIVE	01 03 00 00 00 01 84 0a
13	SEND	01 03 02 12 34 b5 33
115	RECEIVE	01 03 00 00 00 01 84 0a
15	SEND	01 03 02 12 34 b5 33
113	RECEIVE	01 03 00 00 00 01 84 0a
17	SEND	01 03 02 12 34 b5 33
110	RECEIVE	01 03 00 00 00 01 84 0a
20	SEND	01 03 02 12 34 b5 33
108	RECEIVE	01 03 00 00 00 01 84 0a
12	SEND	01 03 02 12 34 b5 33
116	RECEIVE	01 03 00 00 00 01 84 0a
14	SEND	01 03 02 12 34 b5 33
114	RECEIVE	01 03 00 00 00 01 84 0a
16	SEND	01 03 02 12 34 b5 33
111	RECEIVE	01 03 00 00 00 01 84 0a
19	SEND	01 03 02 12 34 b5 33
109	RECEIVE	01 03 00 00 00 01 84 0a

The first column is the delay in milliseconds from the last packet, the second column is the direction of the packet (received from or transmitted to), the last column is the contents of the packet in hexadecimal format. Only the serial ModBUS stream is displayed.
The Traffic Monitor shows all packets received from the serial line, for example if it is a serial slave with an incorrect Modbus response:

3870	SEND	01 03 00 00 00 0a c5 cd
130	RECEIVE	fe fe ff df bc cf bc 9e cf f0 3e 7c bc bc ce 3e cf ce 3c df 8e 8f cf ee ce ce ce bc ce c7 c7 87 be 9e bc bc 9f 3e 3c bc bc 3e bc 8e c7 3c cf 9f be ef bc 01 03 14 42 00 08 7c 00 0b 00 01 00 01 00 00 04 00 c3 48 00 00 44 22 b8 5d

The Traffic Monitor will also display defective packets in yellow (for example a serial master with wrong baud rate):

18	SEND	01 03 02 12 34 b5 33
988	RECEIVE	01 03 00 00 00 01 84 0a
12	SEND	01 03 02 12 34 b5 33
20990	INVALID RECEIVE	20 e0 20 e0 20 e0 20 e0
14994	INVALID RECEIVE	20 e0 20 e0 20 e0 20 e0
14100	INVALID RECEIVE	20 e0 20 e0 20 e0 20 e0
14897	INVALID RECEIVE	20 e0 20 e0 20 e0 20 e0

11. WRITING FROM CLOUD TO DEVICE

11.1. *WRITE TAGS FROM THE CLOUD TO THE DEVICE VIA MQTT (GENERIC CLOUD)*

Through the MQTT configuration, you can write tags in two basic modes:
In the first, the tag name does not appear in the payload, in the second, the tag name is made explicit in the payload.

To write a tag without making its name explicit in the payload, you must subscribe to the topic:

seneca/<ClientID>/info/#

where <*ClientID*> *is the configured client ID*

A publish with topic will then be received from the device:

seneca/<ClientID>/info/<nome tag>

and payload.

{"val": <valore tag>}

or

{"value": <valore tag>}

For example:

making the publish to the topic:

seneca/<ClientID>/info/Pippo

with payload:

{"val": 1234}

The decimal value 1234 is written in the Tag named "Pippo" (be careful with case sensitivity).

To write a tag explicitly stating the name in the payload, you need to subscribe to the topic:

seneca/<*ClientID*> /info

A publish with topic will then be received from the device:

seneca/<*ClientID*> /info
and payload.

{"tags": [{"<nome tag>": <valore tag>}]}

For example:

{"tags": [{"Pippo_fp": 123.46}]}

Writes the floating point value 123.46 in the tag "Pippo_fp"

Or it is possible instead of defining the tag name to use the ID (number that appears in the Tag Vid column (see Tag setup configuration web page):

{"tags_id": [{"<(vid+1)>": <valore tag>}]}

For example:

{"tags_id": [{"25": 789}]}

Writes in the tag with vid = 24 the decimal integer value 789

It is also possible to write more than one tag at the same time with the syntaxes:

{"tags": [{"<nome tag1>": <valore tag1>}, {"<nome tag2>": <valore tag2>},….] }

Or:

{"tags_id": [{"<(vid tag1)+1>": <valore tag1>}, {"< (vid tag2)+1>": <valore tag2>},….] }

For example:

{"tags": [{"Pippo": 1234}, {"Pippo_fp": 123.46}]}
{"tags_id": [{"25": 1234}, {"26": 123.46}]}

They write both tags at the same time.

12. RESETTING THE DEVICE TO ITS FACTORY CONFIGURATION

The factory configuration resets all parameters to default.

To reset the device to the factory configuration it is necessary to follow the procedure below:

Z-KEY-C / Z-KEY-2ETH-C:

1) Remove power from the device
2) Turn dip switches 1 and 2 to ON
3) Power up the device and wait at least 10 seconds
4) Remove power from the device
5) Turn dip switches 1 and 2 to OFF
6) At the next restart the device will have loaded the factory configuration

R-KEY-LT-C:

1) Remove power from the device
2) Set dip switches 1 and 2 of SW2 to ON
3) Power up the device and wait at least 10 seconds
4) Remove power from the device
5) Turn 2 SW2 dip switches to OFF.
6) At the next restart the device will have loaded the factory configuration

13. TEMPLATE EXCEL

A Microsoft Excel™ template is available to create a .bin file to import into the gateway or vice versa. The model can be freely downloaded from the Seneca website.

MODBUS TCP/IP			SERIAL MODBUS RTU				
TAG NR	GATEWAY TAG NAME	GATEWAY MODBUS TCP/IP REGISTER ADDRESS	TARGET MODBUS RTU REGISTER TYPE	TARGET MODBUS RTU DATA TYPE	TARGET CONNECTED TO SERIAL PORT NR	TARGET MODBUS RTU START REGISTER	TARGET MODBUS RTU SLAVE ADDRESS
1	TAG1	1	HOLDING REGISTER	UINT16	#1	3	2
2	TAG2	2	HOLDING REGISTER	UINT16	#1	4	2
3	TAG3	3	HOLDING REGISTER	UINT16	#1	5	2
4	TAG4	5	HOLDING REGISTER	UINT16	#1	6	2
5	TAG5	7	HOLDING REGISTER	UINT16	#1	7	2
6	TAG6	8	HOLDING REGISTER	UINT16	#1	8	2
7	TAG7	9	HOLDING REGISTER	UINT16	#1	9	2
8	TAG8	10	HOLDING REGISTER	UINT16	#1	10	2
9	TAG9	1	COIL	BIT	#1	1	3
10	TAG10	2	COIL	BIT	#1	2	3
11	TAG11	3	COIL	BIT	#1	3	3
12	TAG12	4	COIL	BIT	#1	4	3
13	TAG13	5	COIL	BIT	#1	5	3
14	TAG14	6	COIL	BIT	#1	6	3
15	TAG15	7	COIL	BIT	#1	7	3
16	TAG16	8	COIL	BIT	#1	8	3
17	TAG17	14	HOLDING REGISTER	INT16	#1	13	4
18	TAG18	15	HOLDING REGISTER	INT16	#1	14	4
19	TAG19	16	HOLDING REGISTER	INT16	#1	15	4
20	TAG20	17	HOLDING REGISTER	INT16	#1	16	4
21	TAG21	1	DISCRETE INPUT	BIT	#1	1	5
22	TAG22	2	DISCRETE INPUT	BIT	#1	2	5
23	TAG23	3	DISCRETE INPUT	BIT	#1	3	5

Export CGI file...
Import CGI file...
SENECA Z-KEY TAGS TEMPLATE FOR GATEWAY MODE. E

14. INSTALLING MULTIPLE DEVICES IN A NETWORK USING THE "DHCP FAIL ADDRESS".

When the Gateway is configured with DHCP enabled but does not receive the DHCP server configuration within 2 minutes then it assumes a fail address.
This fail address is 169.254.x.y where x.y are the last two values from the MAC address.
In this way, if you force all devices to DHCP, you can install on the network even if there is no active DHCP server.
When the fail address has been activated (the relative LED stops flashing), you can launch the "Seneca Discovery Device" software and force the preferred IP address to all devices.

15. THE DB9 RS232 CABLE

The DB9 CABLE RS232 CABLE can be obtained from Seneca (it can also be purchased from the e-commerce website www.seneca.it) for connection with a DB9 RS232 device.

16. SUPPORTED MODBUS COMMUNICATION PROTOCOLS

The Modbus communication protocols supported are:

- Modbus RTU/ASCII master/slave (from #1 and #2 serial ports)
- Modbus TCP-IP Client (from the Ethernet port), up to 10 remote TCP-IP Modbus Servers

For more information on these protocols, see the website:
http://www.modbus.org/specs.php.

16.1. SUPPORTED MODBUS FUNCTION CODES

The following Modbus functions are supported:

- Read Coils (function 1)
- Read Discrete Inputs (function 2)
- Read Holding Registers (function 3)
- Read Input Registers (function 4)
- Write Single Coil (function 5)
- Write Single Register (function 6)
- Write multiple Coils (function 15)
- Write Multiple Registers (function 16)

ATTENTION!

All 32-bit variables are contained in 2 consecutive Modbus registers
All 64-bit variables are contained in 4 consecutive Modbus registers

17. INFORMATION ABOUT MODBUS REGISTERS

The following abbreviations are used in the following chapter:

MS	Most Significant
LS	Least Significant
MSBIT	Most Significant Bit
LSBIT	Least Significant Bit
MMSW	"Most" Most Significant Word (16bit)
MSW	Most Significant Word (16bit)
LSW	Least Significant Word (16bit)
LLSW	"Least" Least Significant Word (16bit)
RO	Read Only
UNSIGNED 16 BIT	Unsigned integer register that can assume values from 0 to 65535
SIGNED 16 BIT	Signed integer register that can take values from -32768 to +32767
UNSIGNED 32 BIT	Unsigned integer register that can assume values from 0 to 4294967296
SIGNED 32 BIT	Signed integer register that can take values from -2147483648 to 2147483647
UNSIGNED 64 BIT	Unsigned integer register that can assume values from 0 to 18446744073709551615
SIGNED 64 BIT	Signed integer register that can assume values from -2^63 to 2^63-1
FLOAT 32 BIT	32-bit, single-precision floating-point register (IEEE 754) https://en.wikipedia.org/wiki/IEEE_754
BIT	Boolean register, which can take the values 0 (false) or 1 (true)

17.1. NUMBERING OF "0-BASED" OR "1-BASED" MODBUS ADDRESSES

According to the Modbus standard the Holding Registers are addressable from 0 to 65535, there are 2 different conventions for numbering the addresses: "0-BASED" and "1-BASED".
For greater clarity, Seneca shows its register tables in both conventions.

ATTENTION!

CAREFULLY READ THE DOCUMENTATION OF THE MODBUS MASTER DEVICE IN ORDER TO UNDERSTAND WHICH OF THE TWO CONVENTIONS THE MANUFACTURER HAS DECIDED TO USE

SENECA USES THE "1 BASED" CONVENTION FOR ITS PRODUCTS

17.2. NUMBERING OF MODBUS ADDRESSES WITH "0-BASED" CONVENTION

The numbering is:

HOLDING REGISTER MODBUS ADDRESS (OFFSET)	***MEANING***
0	FIRST REGISTER
1	SECOND REGISTER
2	THIRD REGISTER
3	FOURTH REGISTER
4	FIFTH REGISTER

Therefore, the first register is at address 0.
In the following tables, this convention is indicated with ***"ADDRESS OFFSET".***

17.3. NUMBERING OF MODBUS ADDRESSES WITH "1 BASED" CONVENTION (STANDARD)

The numbering is that established by the Modbus consortium and is of the type:

HOLDING REGISTER MODBUS ADDRESS 4x	***MEANING***
40001	FIRST REGISTER
40002	SECOND REGISTER
40003	THIRD REGISTER
40004	FOURTH REGISTER
40005	FIFTH REGISTER

This convention is indicated with ***"ADDRESS 4x"*** since a 40000 is added to the address so that the first Modbus register is 40001.
A further convention is also possible where the number 4 is omitted in front of the register address:

HOLDING MODBUS ADDRESS WITHOUT 4x	***MEANING***
1	FIRST REGISTER
2	SECOND REGISTER
3	THIRD REGISTER
4	FOURTH REGISTER
5	FIFTH REGISTER

17.4. BIT CONVENTION WITHIN A MODBUS HOLDING REGISTER

A Modbus Holding Register consists of 16 bits with the following convention:

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

For instance, if the value of the register in decimal is
12300
the value 12300 in hexadecimal is:
0x300C

the hexadecimal 0x300C in binary value is:
11 0000 0000 1100

So, using the above convention, we get:

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	0	1	1	0	0	0	0	0	0	0	0	1	1	0	0

17.5. MSB AND LSB BYTE CONVENTION WITHIN A MODBUS HOLDING REGISTER

A Modbus Holding Register consists of 16 bits with the following convention:

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0

LSB Byte (Least Significant Byte) defines the 8 bits ranging from Bit 0 to Bit 7 included, we define MSB Byte (Most Significant Byte) the 8 bits ranging from Bit 8 to Bit 15 inclusive:

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
BYTE MSB								BYTE LSB							

17.6. REPRESENTATION OF A 32-BIT VALUE IN TWO CONSECUTIVE MODBUS HOLDING REGISTERS

The representation of a 32-bit value in the Modbus Holding Registers is made using 2 consecutive Holding Registers (a Holding Register is a 16-bit register). To obtain the 32-bit value it is therefore necessary to read two consecutive registers:
For example, if register 40064 contains the 16 most significant bits (MSW) while register 40065 contains the least significant 16 bits (LSW), the 32-bit value is obtained by composing the 2 registers:

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
40064 MOST SIGNIFICANT WORD															

BIT 15	BIT 14	BIT 13	BIT 12	BIT 11	BIT 10	BIT 9	BIT 8	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
40065 LEAST SIGNIFICANT WORD															

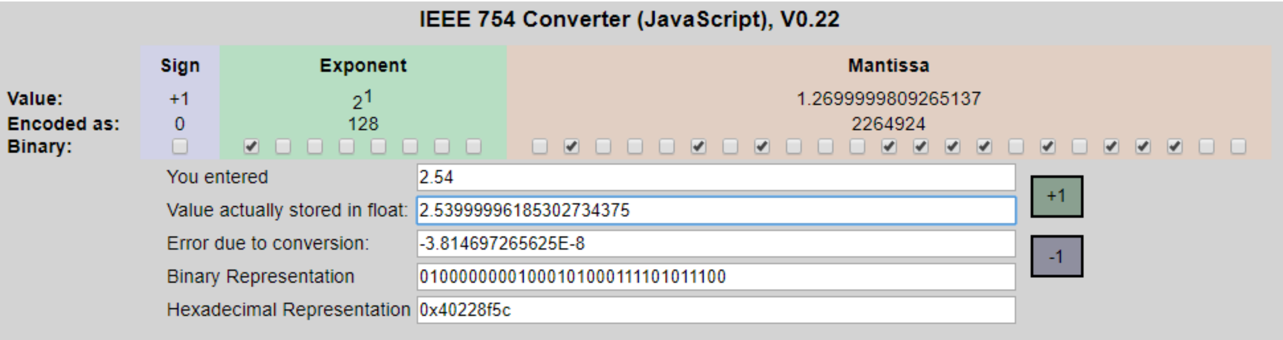
$$Value_{32bit} = Register_{LSW} + (Register_{MSW} * 65536)$$

In the reading registers it is possible to swap the most significant word with the least significant word, therefore it is possible to obtain 40064 as LSW and 40065 as MSW.

17.7. TYPE OF 32-BIT FLOATING POINT DATA (IEEE 754)

The IEEE 754 standard (https://en.wikipedia.org/wiki/IEEE_754) defines the format for representing floating point numbers.
As already mentioned, since it is a 32-bit data type, its representation occupies two 16-bit holding registers.
To obtain a binary/hexadecimal conversion of a floating point value it is possible to refer to an online converter at this address:

http://www.h-schmidt.net/FloatConverter/IEEE754.html



Using the last representation the value 2.54 is represented at 32 bits as:

0x4022 8F5C
Since we have 16-bit registers available, the value must be divided into MSW and LSW:

0x4022 (16418 decimal) are the 16 most significant bits (MSW) while 0x8F5C (36700 decimal) are the 16 least significant bits (LSW).