DIGIRAIL IoT

USER GUIDE V1.1x

novus
We Measure, We Control, We Record

⚠	**Recommended for devices with firmware version V 1.1x and higher.**

ANATEL
09260-20-07089

CE

FC

1 SAFETY ALERTS 5
2 PRESENTATION 6
3 IDENTIFICATION 7
3.1 DEVICE OVERVIEW 7
3.2 DEVICE IDENTIFICATION 7
3.3 DEVICE MODEL 7
4 INSTALLATION 8
4.1 MECHANICAL INSTALLATION 8
4.1.1 DIMENSION 9
4.2 ELECTRICAL INSTALLATION 10
4.2.1 POWER SUPPLY 10
4.2.2 DIGITAL INPUT 11
4.2.3 ANALOG INPUT 11
4.2.4 DIGITAL OUTPUT 12
4.3 LED INDICATORS 12
5 COMMUNICATION INTERFACES 13
5.1 USB INTERFACE 13
5.2 RS485 INTERFACE 13
5.3 ETHERNET INTERFACE 14
5.4 WI-FI INTERFACE 14
6 MQTT PROTOCOL 15
6.1 PUBLICATION AND SUBSCRIPTION TOPICS 15
6.2 PUBLISH BASIC MODEL 15
6.3 TRANSMISSION MODEL FOR DATA AND EVENTS 15
6.3.1 DATA AND EVENTS 15
6.3.2 CHANNEL DATA 15
6.3.3 EVENTS 16
6.4 CONFIGURATION 16
6.4.1 TRANSMITTION MODEL FOR CONFIGURATIONS AND COMMANDS 17
6.5 COMMANDS 17
6.5.1 OUTPUT 17
6.5.2 RESET COUNTERS 18
6.5.3 SET COUNTERS 19
6.5.4 GET DIAGNOSTIC 20
6.5.5 GATEWAY MQTT RS485 22
6.5.6 RESET DIAGNOSTIC 23
6.5.7 LOGS 23
6.5.8 LOGS_PARSED 24
7 MODBUS-TCP PROTOCOL 26
7.1 COMMANDS 26
7.2 REGISTERS TABLE 26
8 CONFIGURATION SOFTWARE 34
8.1 CONFIGURING DIGIRAIL IOT WITH NXPERIENCE 34
8.1.1 GENERAL SETTINGS 34
8.1.2 COMMUNICATION 35
8.1.3 CHANNELS 39
8.1.4 LOGS 41
8.1.5 DIAGRAM 42
8.2 DIAGNOSTICS 43
8.2.1 INFORMATION 43
8.2.2 INPUTS 43

8.2.3 OUTPUTS 44
8.2.4 CONNECTIVITY 44
8.2.5 REMOTE CHANNELS 45
8.2.6 SYSTEM EVENTS 45
9 TECHNICAL SPECIFICATION 46
9.1 CIRCULAR BUFFER AVAILABILITY TABLE 47
9.2 WIRELESS CONNECTIVITY 47
9.3 CERTIFICATION 48
10 WARRANTY 49
11 APPENDIX 1 – RECOMMENDATIONS FOR INSTALLATION IN INDUSTRIAL ENVIRONMENTS 50
11.1 PURPOSE 50
11.2 BEST PRACTICES FOR INDUSTRIAL INSTALLATION 50
11.3 INSTALLATION RECOMMENDATIONS FOR DIGIRAIL IOT DIGITAL INPUT SIGNALS 50
11.3.1 ISOLATED GROUND POWER SUPPLY 50
11.3.2 PULL-UP RESISTORS FOR THE SENSORS 51
11.3.3 HOW TO GROUND THE NEGATIVE TERMINAL OF THE POWER SUPPLY 51
11.3.4 GROUNDED CONDUIT 52
12 APPENDIX 2 – MODBUS-TCP PROTOCOL 53
12.1 INTRODUCTION 53
12.2 REGISTERS 53
12.1.1 COMMANDS 53
12.1.2 TABLE OF REGISTERS: STATUS 53
12.2.1 TABLE OF REGISTERS: CONFIGURATION 61
12.3 ACCESS TO CIRCULAR BUFFER 86
12.3.1 CIRCULAR BUFFER: REGISTER TABLE 86
12.3.2 CIRCULAR BUFFER: AVAILABILITY TABLE 87
12.3.3 EXAMPLE CODE 87
12.3.4 CIRCULAR BUFFER: EXAMPLES 89
12.3.5 COLLECTION: EXAMPLES 90
13 APPENDIX 3 – MQTT PROTOCOL 92
13.1 INTRODUCTION 92
13.2 PUBLISH AND SUBSCRIPE TOPICS 92
13.2.1 PUBLISH BASIC MODEL 92
13.2.2 DATE AND EVENT PUBLISHING MODEL 92
13.2.3 CONFIGURATION AND COMMAND PUBLISHG MODEL 92
13.3 DATA AND EVENTS 93
13.3.1 CHANNEL DATA 93
13.3.2 EVENTS 93
13.4 CONFIGURATIONS 94
13.4.1 HOW TO CHANGE THE CONFIGURATION PARAMETERS 94
13.4.2 HOW TO HANDLE ERRORS WHEN CHANGING THE CONFIGURATION 95
13.4.3 HOW TO CONSULT THE CONFIGURATION PARAMETERS 95
13.4.4 HOW TO HANDLE ERRORS WHEN QUERYING THE CONFIGURATION 96
13.4.5 CONFIGURATION ITEMS 96
13.5 COMMANDS 100
13.5.1 OUTPUT 100
13.5.2 RESET COUNTERS 100
13.5.3 SET COUNTERS 101
13.5.4 GATEWAY MQTT RS485 101
13.5.5 GET DIAGNOSTIC 102
13.5.6 RESET DIAGNOSTIC 103
13.5.7 LOGS 103

13.5.8 LOGS_PARSED 104
13.6 TOPICS IN MULTIPLE CLOUDS 105
13.6.1 AWS 105
13.6.2 GOOGLE IOT 106
13.6.3 MICROSOFT AZURE 106
13.6.4 NOVUS CLOUD 106
13.6.5 GENERIC BROKER 106
13.7 CONFIGURATION VARIABLES 107

1 SAFETY ALERTS

The symbols below are used in the device and throughout this manual to draw the user's attention to important information related to device safety and use.

		
CAUTION Read the manual fully before installing and operating the device.	**CAUTION OR HAZARD** Risk of electric shock.	**ATTENTION** Material sensitive to static charge. Check precautions before handling.

All safety recommendations appearing in this manual must be followed to ensure personal safety and prevent damage to the instrument or system. If the instrument is used in a manner other than that specified in this manual, the device's safety protections may not be effective.

2 PRESENTATION

DigiRail IoT is the ideal tool to read the sensors that monitor the operation of machines, devices, or processes. Among its many applications, this multi-input module allows you to account operation time and stand-by time and the amount of approved and rejected parts, indicates the need for preventive or corrective maintenance, or monitor operating conditions in general.

The device has 6 digital inputs, 2 analog inputs and 2 digital outputs, RS485 interface, USB interface, Wi-Fi or Ethernet communication interface and is compatible with the main clouds on the market. In addition, it can be integrated with MES, SCADA, and ERP systems.

The figure below shows an example of topology for **DigiRail IoT**:

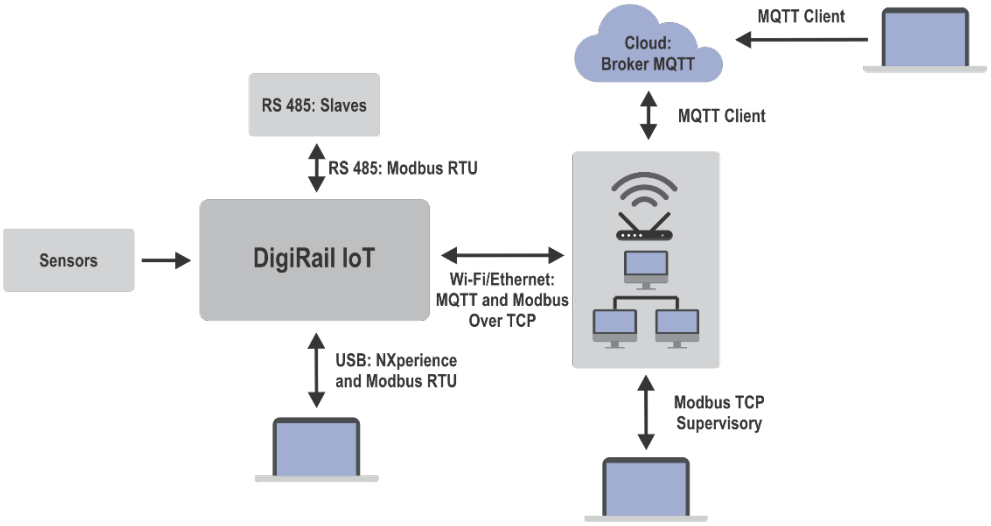

Figure 1

3 IDENTIFICATION

3.1 DEVICE OVERVIEW

Built in ABS+PC and with IP20 protection rating, **DigiRail IoT** has high quality housing, three LEDs on its front and a protection cover with detachable faces to pass the sensors, as shown in the figure below:

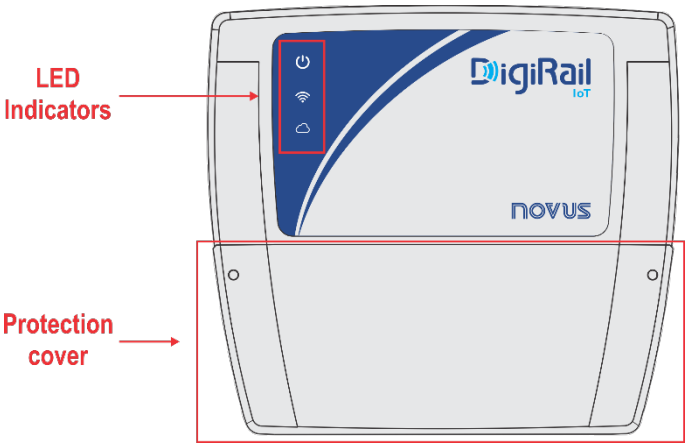

Figure 2

3.2 DEVICE IDENTIFICATION

The identification of the device model is described on the label attached to the back of the housing. This label also provides information on the power supply, MAC address and serial number, as shown in the figure below:

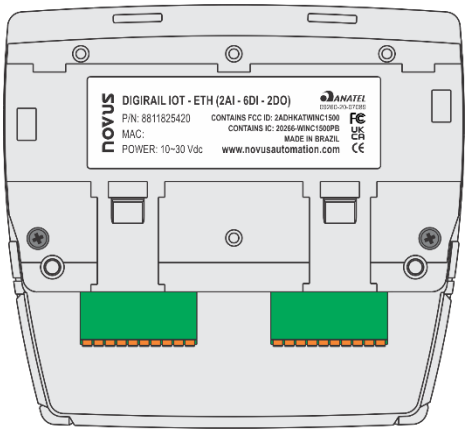

Figure 3

3.3 DEVICE MODEL

DigiRail IoT has two models: **DigiRail IoT – WRL** and **DigiRail IoT – ETH**. Its features are described below:

	Digital Input	Analog Input	Digital Output	USB Interface	RS485 Communication Interface	Ethernet Communication Interface	Wireless Communication Interface
WRL	6	2	2	1	1	×	1
ETH	6	2	2	1	1	1	×

Table 1

4 INSTALLATION

4.1 MECHANICAL INSTALLATION

As shown in the figure below, **DigiRail IoT** can be installed on DIN 35 mm rail. You must fix it with its back clips:

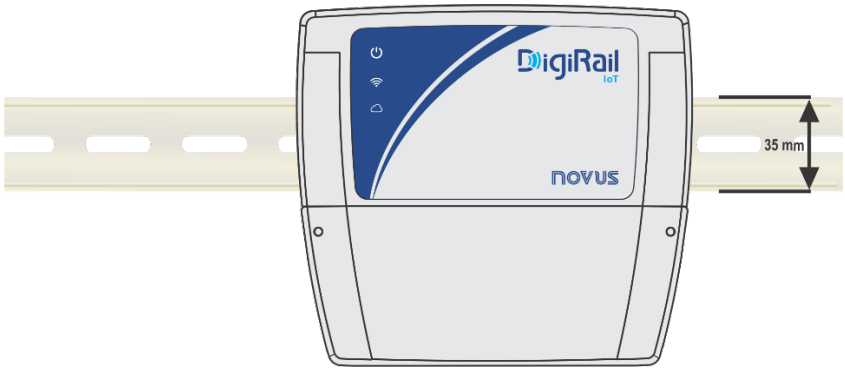

Figure 4

In addition, the device also has two holes to fix it with screws, as shown in the figure below:

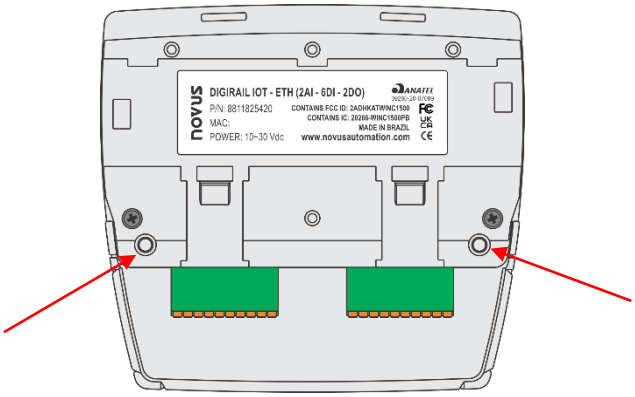

Figure 5

DigiRail IoT has a removable protection cover to protect its connection terminals. The protection cover has three detachable areas, one at the bottom and one at each side, so you can easily handle the sensors:

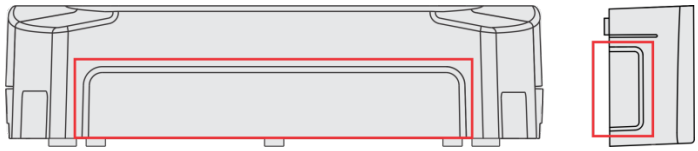

Figure 6

The protection cover has two pins, located on the sides of the housing, which help you fit it into the device. Once the cover has been installed, you will need a screwdriver to remove it.

4.1.1 DIMENSION

DigiRail IoT has the following dimensions:

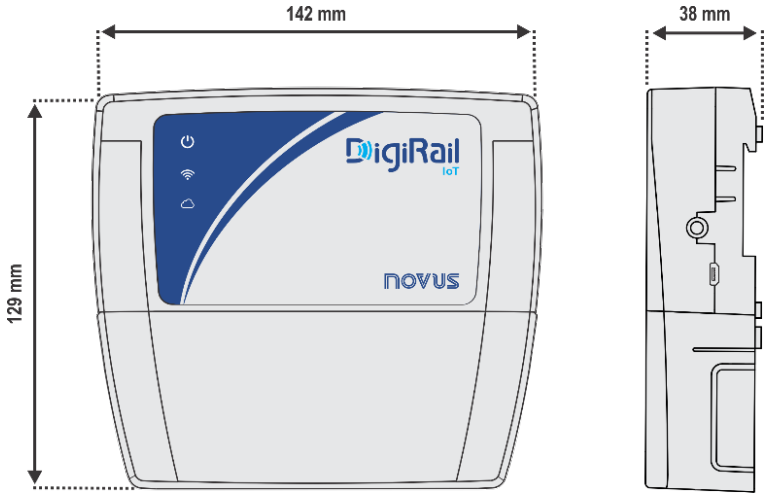

Figure 7

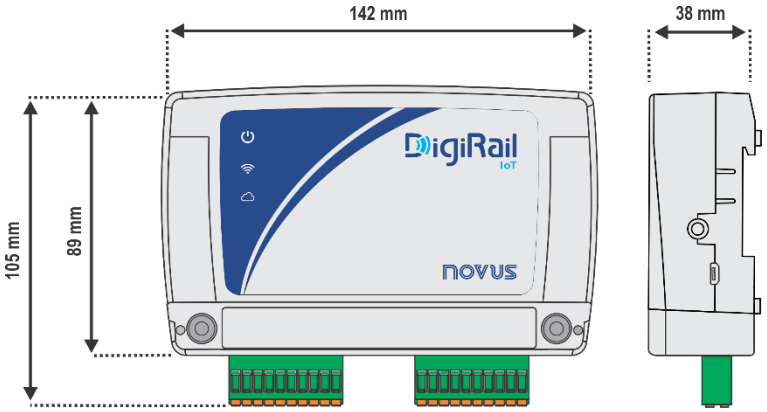

Figure 8

DigiRail IoT protection cover has the following dimensions:

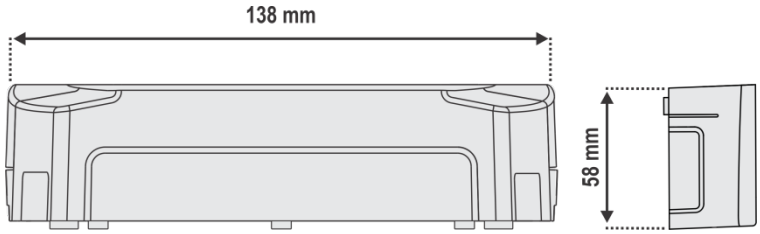

Figure 9

4.2 ELECTRICAL INSTALLATION

DigiRail IoT has three detachable connection terminals to connect the external power supply, RS485, digital inputs and outputs and analog inputs, as shown in the figure below:

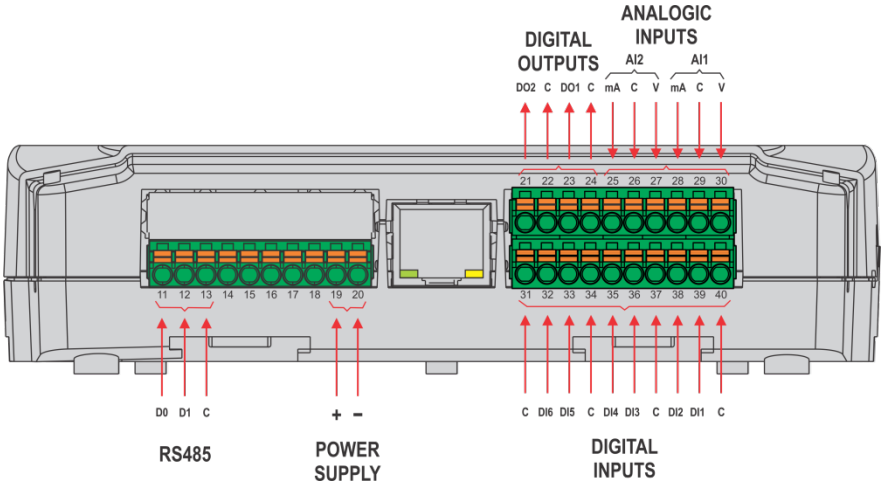

Figure 10

To connect the sensors, you must previously remove the connection terminals from the device and observe the enumeration recorded in the housing, as shown in the figure of electrical connections above.

	**Inputs, outputs and communication interfaces of this device are not isolated from the power supply or from each other.**

INSTALLATION RECOMMENDATIONS

- Electronic and analog signal conductors must run through the plant separately from the output and power conductors. If possible, in grounded conduits.
- The power supply for the electronic instruments must come from a proper electrical grid for instrumentation.
- It is recommended to use RC FILTERS (noise suppressors) in contactor coils, solenoids, etc.
- In control applications, it is essential to consider what can happen when any part of the system fails. The device internal security features do not guarantee full protection.
- The electrical connections must be made with the device connection terminals. Before connecting them, make sure that the connections have been made properly.

4.2.1 POWER SUPPLY

The power supply connection is made at the terminals, according to the figure below. You must use a power supply with voltage between 10 and 30 V. You can also use 12 and 24 Vdc power supplies.

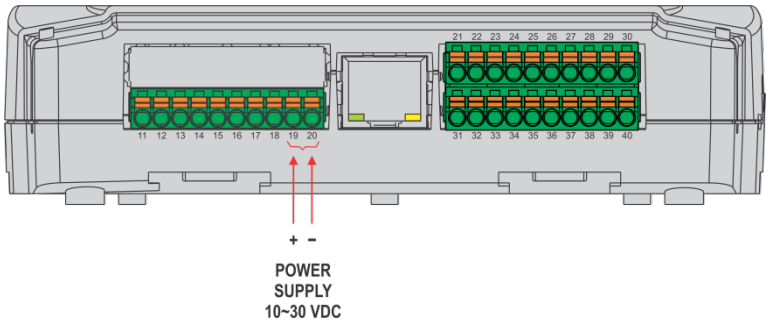

Figure 11

4.2.2 DIGITAL INPUT

DigiRail IoT has digital input channels that can be configured in "Counting" or "Event" modes. Regardless of the selected function, you must configure the type of sensor connected to the input: PNP, NPN or Dry Contact. After that, select the edge of the digital signal to generate the count or event: Rising edge, falling edge or both edges.

CORRELATION BETWEEN SENSOR TYPE, SENSOR STATUS AND LOGICAL LEVEL		
Sensor Type	**Sensor Status**	**Logical Level**
PNP	Open	0
	Closed	1
NPN	Open	1
	Closed	0
Dry Contact	Open	1
	Closed	0

Table 2

The digital input connection is made at the corresponding terminals, as shown in the figure below:

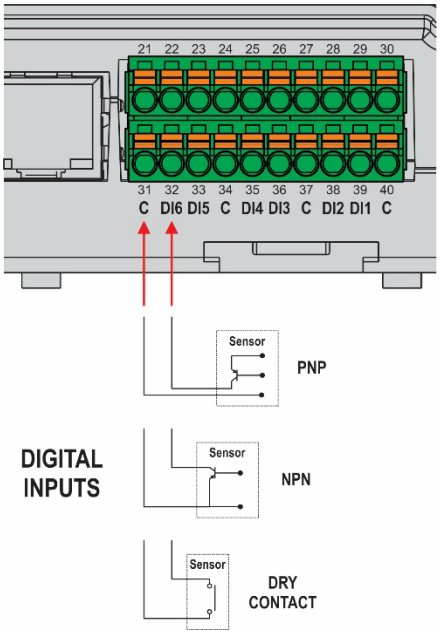

Figure 12

4.2.3 ANALOG INPUT

The analog input connection is made at the corresponding terminals, as shown in the figure below:

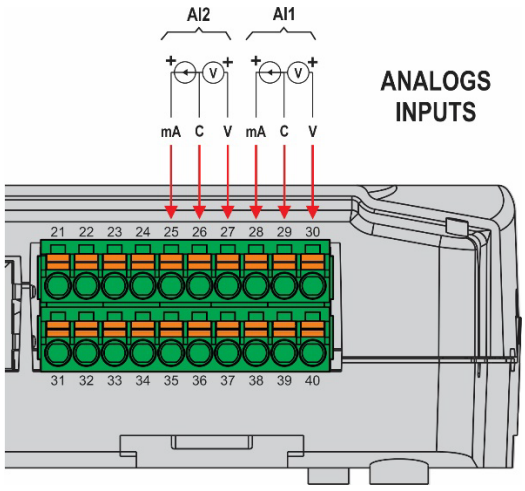

Figure 13

4.2.4 DIGITAL OUTPUT

The digital output connection is made at the corresponding terminals, as shown in the figure below:

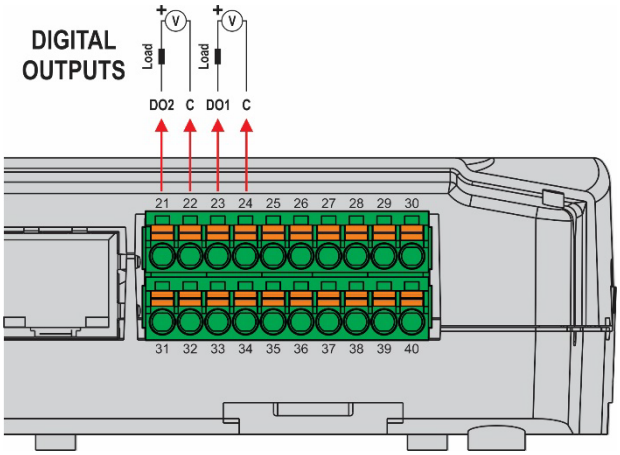

Figure 14

4.3 LED INDICATORS

DigiRail IoT has three LEDs, located on the front of the device, as shown in the figure below:

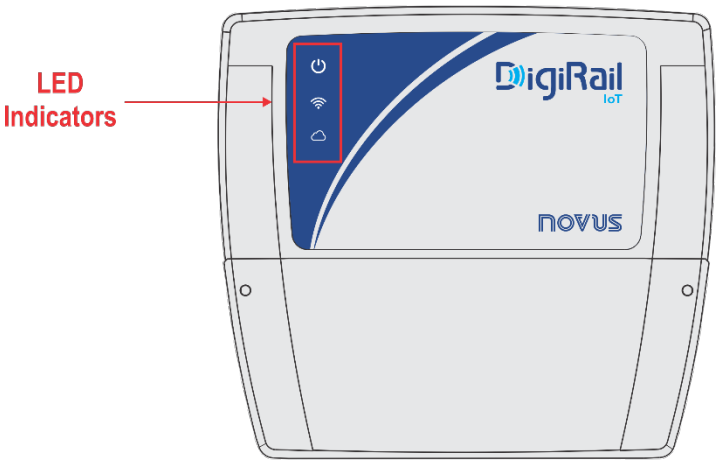

Figure 15

The operation and description of each LED are as follows:

NAME	SYMBOL	STATUS	DESCRIPTION
STATUS		Off	Device off.
		On	Device on.
		Flashing	Device in firmware update mode.
WI-FI / ETHERNET CONNECTION INDICATOR		On	The connection has been established.
		Flashing	The data is being transmitted.
		Off	The connection has not been established.
MQTT BROKER CONNECTION INDICATOR		On	The connection has been established.
		Flashing	The data is being transmitted.
		Off	The connection is disabled or failed when initializing.

Table 3

5 COMMUNICATION INTERFACES

5.1 USB INTERFACE

DigiRail IoT has a USB port, located on the side of the housing, to configure and perform the device C. You must use a USB cable in the standard micro-USB (not supplied) to connect the device with a desktop or notebook.

When installing the **NXperience** configuration software, the USB port drivers will be automatically installed (see chapter CONFIGURATION SOFTWARE).

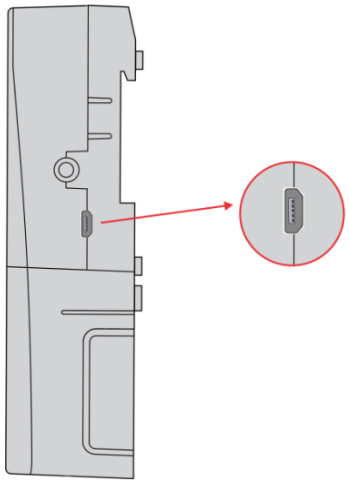

Figure 16

	**The USB interface is NOT ISOLATED.** **It should be used temporarily to CONFIGURE or PERFORM THE DIAGNOSIS of the device.**

5.2 RS485 INTERFACE

The RS485 interface can be configured to operate in 3 different modes: **1)** Gateway from Modbus-TCP to Modbus-RTU, **2)** Master mode of a Modbus-RTU or **3)** Slave mode of a Modbus-RTU network.

The RS485 connection interface is located on one of the **DigiRail IoT** detachable terminals, as shown in the figure below:

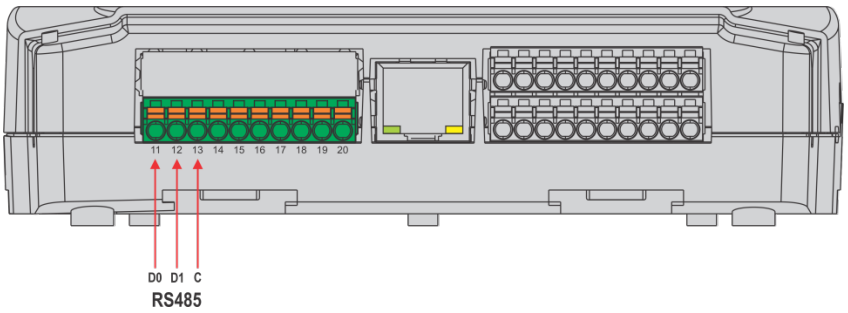

Figure 17

The RS485 interface can be configured to operate at the following speeds (Baud Rates): 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200. Besides, it can be configured to operate with 1 or 2 Stop Bits and in even parity, odd parity, and no parity. You can configure all these parameters using the **NXperience** software (see the CONFIGURATION SOFTWARE chapter).

	**The RS485 interface only works when the DigiRail IoT is connected to an external power supply. It will not operate when the device is being powered only by the USB interface.** **The device has an internal 120 ohms termination resistor for the RS485 interface.**

The table below shows how to connect the connectors to the RS485 communication interface:

D0	$\overline{D}$	D-	A	Inverted bidirectional data line.	**Terminal 11**
D1	D	D+	B	Bidirectional data line.	**Terminal 12**
C				Optional connection which improves the communication performance.	**Terminal 13**
GND					

Table 4

You can find more details about implementing a device network via RS485 in the document "RS485 and RS422 basics", available at www.novusautomation.com.

5.3 ETHERNET INTERFACE

DigiRail IoT – ETH has an Ethernet interface, located next to the device terminals, as shown in the figure below:

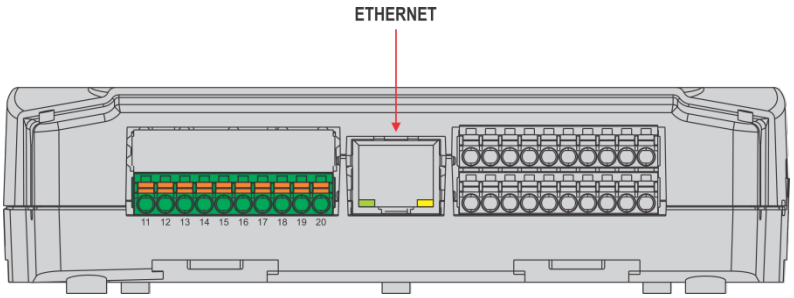

Figure 18

If the Ethernet interface is enabled and the device is connected to an Ethernet network, the LED on the front of the device remains on. This LED will remain on and flashing if data is being sent over this interface.

5.4 WI-FI INTERFACE

DigiRail IoT – WRL has an 802.11 Wi-Fi interface in b/g/n 2.4 GHz standards, operating through an internal antenna. This interface supports WPA-Personal (PSK) WPA/WPA2 TKIP/AES/TKIP and AES encryption.

If the Wi-Fi interface is enabled and the device is connected to a Wi-Fi network, the LED on the front of the device remains on. This LED will remain on and flashing if data is being sent over this interface.

6 MQTT PROTOCOL

DigiRail IoT is compatible with Message Queue Telemetry Transport (MQTT) protocol, 3.1 and 3.1.1 versions, which allows publishing data in the cloud, and supports the following MQTT Brokers: Google Cloud, Microsoft Azure, AWS, **NOVUS Cloud**, and generic MQTT Brokers.

This chapter describes the structure of the data published in the cloud and introduces the structure to send settings to the device.

6.1 PUBLICATION AND SUBSCRIPTION TOPICS

As described below, **DigiRail IoT** uses five topics for operation:

- **Topic to publish periodic data and events:** Used to publish data generated on the device, i.e., the logs. There are two types: **`channel`** and **`events`**.
- **Topic to receive the configuration:** Used to receive configuration data. The device subscribes to this topic to receive configuration data. For each configuration received, a confirmation reply is published in the configuration confirmation topic.
- **Topic to confirm the configuration:** The device publishes the current configuration in this topic. Every time a configuration is received, the device publishes a confirmation in this topic. After a configuration is applied to the device, the current configuration is also published in this topic.
- **Topic to receive commands:** Used to receive commands. The device subscribes to this topic to receive commands and signals the execution of a command by publishing in the command confirmation topic.
- **Topic to confirm the command:** The device publishes the result of commands executed in this topic.

Examples of topics for a generic Broker:

TOPIC	USE
Topic to publish periodic data and events	NOVUS/device1/events
Topic to receive the configuration	NOVUS/device1/config
Topic to confirm the configuration	NOVUS/device1/ack/config
Topic to receive commands	NOVUS/device1/command
Topic to confirm commands	NOVUS/device1/ack/command

Table 5

6.2 PUBLISH BASIC MODEL

To simplify the treatment of MQTT message content, publications will always display the product model identifier and the user-defined identifier, labeled by the "**`pid`**" and "**`device_id`**" fields, respectively. The value of the "**`device_id`**" field is set in the Device ID parameter of the MQTT settings of the **NXperience** software.

Applicable **DigiRail IoT** Identifiers:

MODEL	PID
DigiRail IoT ETH	51584019
DigiRail IoT WRL	51518482

Table 6

6.3 TRANSMISSION MODEL FOR DATA AND EVENTS

The publication of events and data generated by the device follows the standard MQTT model and uses a topic defined during configuration.

6.3.1 DATA AND EVENTS

The data will be published in the topic defined for the publication of periodic data and events. The type of data is indicated in the JSON message. For all data, the timestamps used are in Unix timestamp UTC format (GMT 0).

6.3.2 CHANNEL DATA

The channel data is published periodically, according to the device configuration. The data is in JSON format and has the following key/value sets:

```
{
    "pid": 51387408,
    "device_id": "device0",
    "channels" : {
        "timestamp":1585819219,
        "chd1_value":0,
        "chd2_value":0,
        "chd3_value":0,
        "chd4_value":0,
        "chd5_value":0,
        "chd6_value":0,
        "ch1_user_range":2,
        "ch2_user_range":-19991,
```

```
        "chr1_value":0,
        "chr2_value":0,
        "chr3_value":0,
        "chr4_value":0,
        "chr5_value":0,
        "chr6_value":0,
        "chr7_value":0,
        "chr8_value":0
    }
}
```


Notes:

- The **timestamp** value is the timestamp in Unix UTC format at the time the device reads the channel.
- **chdX_value** corresponds to the information of the digital channels at the time of timestamp. If the channel is not enabled, it will not appear in the JSON. If the channel is in "Register" mode, the value will correspond to the logical level of the digital channel at that moment. If the channel is in "Counting" mode, the value will correspond to the counter value at that time.
- **chX_user_range** informs the value of the analog input in the range configured by the user and at the time of the timestamp. If the analog channel is not enabled, it will not appear in the JSON.
- **chrX_user_range** informs the last value read from the slave device configured for the respective remote channel. If the analog channel is not enabled, it will not appear in the JSON.

6.3.3 EVENTS

When the digital channel is configured in "Event" mode and an event occurs, an event type message will be generated, indicating the channel, the timestamp, and the edge where it occurred. The data are in JSON format and have the following key/value sets:


```
{
    "pid": 51387408,
    "device_id": "device0",
    "events": {
        "chd1": {
            "timestamp":1585819219.685,
            "edge":1,
        }
    }
}
```


Notes:

- The **timestamp** value is also in Unix timestamp format in UTC (GMT 0), but the milliseconds of the event have been added as fractional part.
- Regarding the **edge** value: "1" means that the event occurred on a rising edge. "0" means that the event occurred on a falling edge.

6.4 CONFIGURATION

Some sets of device configuration can be changed or consulted via MQTT when publishing in the topic to receive device configuration. A confirmation of this publication is received in the configuration confirmation topic.

The available configuration items for this device type are:

CONFIGURATION ITEM	DESCRIPTION
rtc	RTC (Real Time Clock - device internal clock) configuration.
device	General device configuration.
chdX	Digital channel 'X' configuration (Available: **chd1**, **chd2**, **chd3**, **chd4**, **chd5**, and **chd6**).
Periodic counter reset	Configuration of the digital counters reset periodicity.
chX	Configuration of the analog channel 'X' (Available: **ch1** and **ch2**).
chrX	Configuration of the remote channel 'X' (Available: **chr1**, **chr2**, **chr3**, **chr4**, **chr5**, **chr6**, **chr7**, and **chr8**).
eth	Network configuration parameters.
wifi	Wi-Fi interface configuration (When available).
ntp	NTP server configuration for automatic clock adjust.
modbus-tcp	Modbus-TCP protocol configuration.
rs485	RS485 interface configuration.

Table 7

6.4.1 TRANSMITTION MODEL FOR CONFIGURATIONS AND COMMANDS

The basic operating model of the commands and configurations was developed to allow synchronization of the device settings and conditions with the cloud.

In this model there are two basic concepts:

- **Desired properties:** These are the conditions and configurations that the backend application can change or query on the device with which it interacts.
- **Reported properties:** These are the properties used as a response to receive **`Desired properties`**, where the device reports its status or the result of a command.

This message exchange model needs two different topics to work. The first is the topic in which the device is subscribed to receive the **`Desired properties`**. This step, initiated by the application, is called "**`request`**". The device uses the second topic to publish the **`Reported properties`** after the command or configuration is executed. This step is called "**`response`**".

For details on sending configurations via MQTT to **DigiRail IoT**, check APPENDIX 3 – MQTT PROTOCOL.

6.5 COMMANDS

Following the same model of sending settings, the commands must be published in the **Topic to receive commands**. The type of data is indicated in the JSON message. The return of the command execution is done through the **Topic to confirm the command**.

The available commands for **DigiRail IoT** are:

- **Output:** Used to obtain or modify the digital outputs status.
- **Reset counters:** Used to apply a reset to the digital counters.
- **Set Counters:** Used to change the value of the digital channel counters.
- **Get diagnostic:** Used to obtain diagnostic data from the device.

6.5.1 OUTPUT

This command modifies the device output status.

FORMAT OF THE OUTPUT COMMAND TO MODIFY THE OUTPUT STATUS:

```
{
    "timestamp":1585819219,
    "desired": {
        "output": {
            "out1":1,
            "out2":1
        }
    }
}
```

It is not necessary to publish the status that will not be modified.

FORMAT OF THE COMMAND OUTPUT RESPONSE:

```
{
    "pid": 51387408,
    "device_id": "device0",
    "timestamp":1585819219,
    "reported": {
        "output": {
            "error": 0,
            "out1":1,
            "out2":1
        }
    }
}
```

Notes:

- The **`timestamp`** is the same as the command received (**`desired`**).
- The status described in the **`desired`** step will only be applied if the execution is done without errors.
- The value shown in the **`error`** field is an integer and reports the first error found in the execution of the command, as shown in the error codes table below:

CODE	DESCRIPTION
Error 0	Success.
Error 1	The structure is correct, but the device has received an out-of-range parameter.
Error 2	The structure is correct, but the device has received an unknown parameter.

Table 8

There are, however, unanswered error cases from the device, as shown below:

- The JSON structure is wrong.
- The structure is right, but some element is missing (**timestamp**, **desired**, **item**).

In case of error, none of the parameters will be accepted and the device will not go into configuration mode.

- If the command has failed, the status indicated in **reported** will be the current ones.

This command can also be used to consult the status of the device outputs when sent with the format provided below.

FORMAT OF THE *OUTPUT* COMMAND TO OBTAIN THE OUTPUT STATUS:

```
{
    "timestamp":1585819219,
    "desired": {
        "output": {}
    }
}
```

THE FORMAT OF THE RESPONSE TO OBTAIN THE OUTPUT STATUS IS THE SAME FORMAT AS THE RESPONSE TO THE COMMAND TO MODIFY THEM:

```
{
    "pid": 51387408,
    "device_id": "device0",
    "timestamp":1585819219,
    "reported": {
        "output": {
            "error": 0,
            "out1":1,
            "out2":1
        }
    }
}
```

6.5.2 RESET COUNTERS

The **reset counters** command is used so that the application can reset the digital channels counters. A digital channel needs to have MQTT enabled for it to be restarted through this interface.

The structure used for this command follows the same model as for sending configurations, using the concepts of "**desired**" and "**reported**".

The **reset_chdX** value can assume values of 0 or 1. The value of "1" indicates that a reset is to be applied to the counter of the corresponding digital channel. The value "0" indicates that the counter should not be changed. In this case, it is also possible to simply omit the JSON channel.

REQUEST *RESET COUNTERS*:

```
{
    "timestamp":1585819219,
    "desired": {
        "reset_counters" : {
            "reset_chd2":1,
            "reset_chd4":1
        }
    }
}
```

RESPONSE *RESET COUNTERS*:

```
{
    "pid": 51387408,
    "device_id": "device0",
    "timestamp":1585819219,
    "reported" : {
        "reset_counters": {
           "error": 0,
           "reset_chd1":0,
           "reset_chd2":0,
           "reset_chd3":0,
           "reset_chd4":0,
           "reset_chd5":0,
           "reset_chd6":0
        }
    }
}
```

Notes:

- The **timestamp** is the same as the command received (**desired**).
- The status described in the **desired** step will only be applied if the execution is done without errors.
- The **error** value is an integer and reports the error found during the command execution.

- In this example, digital channels 1, 3, 5 and 6 do not appear in the JSON **desired** since you do not want to reset their counters.

6.5.3 SET COUNTERS

The **set counters** command is used so that the application can change the value of the digital channel counters. You must enable the permission to change the configuration through MQTT to modify a digital channel through this interface.

The structure of this command follows the same model as for sending settings, using the "**desired**" and "**reported**" concepts.

The **set_chdX** can assume any value between 0 and X. If the setting allows it, the channel will immediately assume the defined value when sending the value on the **set_chdX** field. You must omit the channel from the JSON in order not to change the counter value of a channel.

REQUEST *SET COUNTERS*:

```
{
    "timestamp":1620413979
    "desired": {
        "set_counters" : {
                "set_chd2":6500,
               "set_chd3":10
            }
    }
}
```

RESPONSE SET COUNTERS:

```
{
    "pid": 51387408,
    "device_id": "device0",
    "timestamp":1620413979,
    "reported" : {
        "set_counters": {
           "error": 0,
           "set_chd1":0,
           "set_chd2":6500,
           "set_chd3":10,
           "set_chd4":0,
           "set_chd5":0,
           "set_chd6":0
        }
    }
}
```

Notes:

- The **timestamp** is the same as the command received (**desired**).
- The status described in the **desired** step will only be applied if the execution is done without errors.
- The **error** value is an integer and reports the error found during the command execution.
- In this example, digital channels 1, 4, 5 and 6 do not appear in the JSON **desired** since you do not want to change their counters. In the response, the current value of the digital channel will be returned. For digital channels 1, 4, 5 and 6 the current value is assumed to be zero.

6.5.4 GET DIAGNOSTIC

The **`get diagnostic`** command returns diagnostic data from the device.

REQUEST *GET DIAGNOSTIC*:

```
{
    "timestamp":1585819219,
    "desired" : {
        "diag" : {}
    }
}
```

RESPONSE GET DIAGNOSTIC:

```
{
    "pid": 51387408,
    "device_id": "device0",
    "timestamp":1585819219,
    "reported" : {
       "diag": {
            "title": "Pci v2",
            "location":"home",
            "curr_timestamp":1589326517,
            "cfg_timestamp":1589311676,
            "fw_v":"1.23",
            "mqtt_queue":1,
            "sn":"00000001",
            "curr_rssi":"55",
            "min_rssi":"46",
            "max_rssi":"87",
            "avg_rssi":"54",
            "ipv4":[ 192, 168, 0, 23 ]
        }
    }
}
```

If the **Publish Diagnostics Periodically** parameter of the **NXperience** configuration software (see the MQTT PROTOCOL section of the CONFIGURATION SOFTWARE chapter) is enabled, the system event occurrence counters will also be added to the response:

```
{
     "pid_id":51387408,
   "device_id":"device0",
   "timestamp":1585819219,
   "reported":{
      "diag":{
         "error":0,
         "title":"Pci v2",
         "location":" home ",
         "curr_timestamp":1589326517,
         "cfg_timestamp":1589311676,
         "fw_v":"1.23",
         "mqtt_queue":1,
         "sn":"00000001",
         "curr_rssi":"55",
         "min_rssi":"45",
         "max_rssi":"70",
         "avg_rssi":"55",
         "ipv4":[
            192,
            168,
            0,
            23
         ],
         "log_counters":{
            "pwr_on":1,
            "pwr_sw_reset":0,
            "net_disconnected":1,
            "wifi_prov_error":0,
            "dhcp_error":0,
```

```
            "dns_error_1":0,
            "dns_error_2":0,
            "cfg_updated":1,
            "fw_updated":0
         },
         "watchdog_counters":{
            "analog":"0",
            "data_storage":"0",
            "record_storage":"0",
            "digital":"0",
            "modbus":"0",
            "record_periodic":"0",
            "mqtt":"1",
            "network":"0"
         }
      }
   }
}
```


Notes:

- The **`title`** and **`location`** fields are defined in the general configuration frame of the configurator software.
- The **`curr_timestamp`** field presents the current timestamp of the device, i.e., obtained from its internal clock and is in Unix timestamp UTC format.
- The **`cfg_timestamp`** field presents the timestamp of the last configuration applied to the device and is also in Unix timestamp UTC format.
- The **`fw_v`** field presents the firmware version of the device.
- The **`mqtt_queue`** field presents the number of logs pending sending via MQTT.
- The **`sn`** field presents the serial number of the device.
- The **`curr_rssi`** field informs the quality of the Wi-Fi signal, which is measured instantaneously. The value is shown as a percentage. Thus, the higher the value, the better the signal. The **`min_rssi`**, **`max_rssi`**, and **`avg_rssi`** fields complement the diagnosis of the Wi-Fi signal quality, returning the minimum, maximum, and average value, respectively.
- The **`ipv4`** field informs the IP of the device on the network.
- The **`log_counters`** field informs the number of occurrences of each system log event.
- The **`watchdog_counter`** field informs the number of occurrences of each system Watchdog event.

6.5.5 GATEWAY MQTT RS485

Sending packets through the RS485 serial interface via MQTT allows you to read data from a local network (Modbus RTU, for example) and send commands remotely via the MQTT protocol. In this case, **DigiRail IoT** operates as a Gateway, communicating with the slave devices through the RS485 serial interface.

To send commands remotely, it is necessary to connect another MQTT client to the Broker to which **DigiRail IoT** is connected and, in the sequence, register in the topic configured for command confirmation. The command must then be published in the topic configured in **DigiRail IoT** to receive commands.

Modbus RTU commands can be published in hexadecimal format with the following structure:

```
{
    "timestamp":XXXXXX,
    "desired": {
        "gateway_485": {"mb_buffer":"bytes in hexadecimal to transmit through serial 485"}
    }
}
```

Below is an example of a message to be published in the command sending topic:

```
{
    "timestamp":15,
    "desired": {
        "gateway_485": {"mb_buffer":"02 03 00 00 00 0A C5 FE"}
    }
}
```

In sequence, the response received through the RS485 serial interface will be published by **DigiRail IoT** in the topic assigned to the commands confirmation, following the format:

```
{
    "pid": XXX,
    "device_id":XX,
     "timestamp":XXXX,
     "reported": {
        "gateway_485": {"error":0; "mb_buffer":"bytes received in response to the command sent"}
    }
}
```

An example message that could be received in the command confirmation topic:

```
{
    "pid": 51387408,
    "device_id":"DeviceName",
     "timestamp":15,
     "reported": {
        "gateway_485": {"error":0; "mb_buffer":"00 03 14 19 C7 00 00 06 4E 00 00 04 E0 00 00 03 D0 00 00 03 D
0 00 00 1B 13"}
    }
}
```

6.5.6 RESET DIAGNOSTIC

The **`reset diagnostic`** command is used so that the application can reset counters related to internal system events and Wi-Fi signal quality (RSSI) measurement data.

The structure used for this command follows the same model as for sending configurations and uses the concepts of "**`desired`**" and "**`reported`**".

The values of the **`reset_watchdog_counter`**, **`reset_x_counter`**, and **`reset_diag_rssi`** fields can have values of 0 or 1. The value "1" means that a reset is to be applied to the corresponding parameter. The value "0" indicates that the parameter should not be changed. In this case you can also simply omit the JSON channel.

REQUEST *RESET DIAGNOSTIC*:

```
{
    "timestamp":1585819219,
    "desired": {
        "reset_diag": {
            "reset_watchdog_counter":0,
            "reset_logs_counter":1,
            "reset_diag_rssi":1
        }
    }
}
```

RESPONSE *RESET DIAGNOSTIC*:

```
{
    "pid": 51387408,
    "device_id": "device0",
    "timestamp":1585819219,
    "reported": {
        "reset_diag": {
           "error": 0,
           "reset_watchdog_counter":0,
           "reset_logs_counter":0,
           "reset_diag_rssi":0
        }
    }
}
```

Notes:

- The **`timestamp`** is the same as the received command (**`desired`**).
- The status described in **`desired`** will only be applied if execution is done without errors.
- The value of **`error`** is an integer and reports the error encountered during the execution of the command.

6.5.7 LOGS

The logs command returns the last 50 log events from the system. All events will have an ID, which can be queried with this command, and a timestamp of the occurrence. You can see a detailed description of the log in **Table 8**.

REQUEST *LOGS*:

```
{
    "timestamp":1585819219,
    "desired": {
        "logs": {}
    }
}
```

RESPONSE *LOGS*:

```
{
  "pid":51387408,
   "device_id":"digirail_iot",
   "timestamp":1585819219,
   "reported":{
      "logs":{
         "error":0,
         "events":[
            {
               "ts":1638193059,
               "id":9
            },
```

```
            {
               "ts":1638193055,
               "id":10
            },
            {
               "ts":1638192333,
               "id":9
            },
            {
               "ts":1636466491,
               "id":4
            }
         ]
      }
   }
}
```


6.5.8 LOGS_PARSED

Due to device memory limitations, the **`logs_parsed`** command returns only the last 30 system log events. However, instead of giving an ID, there will be a short description of the log, plus the timestamp of the occurrence, like the **`logs`** command. You can see a detailed description of the log in **Table 8**.

REQUEST LOGS_PARSED:


```
{
    "timestamp":1585819219,
    "desired": {
        "logs_parsed": {}
    }
}
```


RESPONSE LOGS_PARSED:


```
{
  "pid": 51387408
   "device_id":"digirail_iot",
   "timestamp":1585819219,
   "reported":{
      "logs_parsed":{
         "error":0,
         "events":[
            {
               "ts":1638193059,
               "mqtt":"connected"
            },
            {
               "ts":1638193055,
               "mqtt":"disconnected"
            },
            {
               "ts":1638192333,
               "mqtt":"connected"
            },
            {
               "ts":1636468024,
               "net":"connected"
            }
         ]
      }
   }
}
```

The table below shows a detailed description of the logs:

CODE	LOGS_PARSED		DESCRIPTION
0	pwr	on	Standard startup.
1	pwr	sw_reset	Startup triggered by software reset.
2	pwr	wdt_reset	Startup triggered by internal Watchdog.
3	pwr	lvd_reset	Startup triggered by power outage.
4	net	connected	Connected to a network (Wi-Fi or Ethernet).
5	net	disconnected	Disconnected from the network (Wi-Fi or Ethernet).
6	wifi	prov_error	Wi-Fi provisioning failure (SSID or password incorrect).
7	dhcp	error	DHCP error.
8	sntp	error	SNTP error.
9	mqtt	connected	Connected to a MQTT broker.
10	mqtt	disconnected	Disconnected from the MQTT broker.
11	mqtt	sub_error	MQTT topics subscription error.
12	mqtt	pub_error	MQTT topics publishing error.
13	mqtt	alter_int	Publish interval has been changed to an alternative interval.
14	mqtt	default_int	Publish interval has been changed to a default interval.
15	dns	error_1	DNS internal error - Phase 1.
16	dns	error_2	DNS internal error - Phase 2.
17	dns	error_3	DNS internal error - Phase 3.
18	mem	init_error	There was an error during the circular buffer initialization. Device has recovered.
19	mem	not_init	The circular buffer has not been initialized.
20	mem	read_error	There was a failure while reading the circular buffer.
21	cfg	updated	Device configuration updated.
22	fw	updated	Device firmware updated.

Table 9

7 MODBUS-TCP PROTOCOL

DigiRail IoT is compatible with the Modbus-TCP protocol, a data communication protocol used to connect the device to supervisory control and data acquisition (SCADA) systems. It supports up to 3 simultaneous connections and allows up to 3 Modbus-TCP clients (masters) to monitor it at the same time. **DigiRail IoT** operates both as a Modbus-TCP server (slave) and as a TCP/RTU gateway.

As a server (slave), it responds to the configured Modbus RTU address. For address that diverge from the configured address value, it will operate as a TCP/RTU gateway. In this case, the packed will be sent to the RS485 interface and, if there is a reply from any Modbus RTU slave, replied to the Modbus-RTU client (master) that generated the request.

For more information about Modbus-TCP protocol, check APPENDIX 2 – MODBUS-TCP PROTOCOL.

7.1 COMMANDS

READ HOLDING REGISTERS – 0x03:

This command can be used to read the value of one or up to a maximum of 125 consecutive registers, according to the table below.

WRITE HOLDING REGISTERS – 0x06:

This command can be used to write in a register, according to the table below.

WRITE MULTIPLE HOLDING REGISTERS – 0x16:

This command can be used to write in multiple registers, according to the table below.

7.2 REGISTERS TABLE

Below is the table of registers supported by the device:

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1	HR_PRODUCT_CODE	Product code.	786	786	RO
2	HR_SERIAL_NUMBER_H	Serial number (32-bits).	0x0000	0xFFFF	RO
3	HR_SERIAL_NUMBER_L		0x0000	0xFFFF	RO
4	HR_FIRMWARE_VERSION	Version firmware x100.	100	65535	RO
		Reserved.			
6	HR_MAC_ADDR_0_1	MAC Address. Hexadecimal format with 2 numbers per register. 0 : 1 : 2 : 3 : 4 : 5	0x0000	0xFFFF	RO
7	HR_MAC_ADDR_2_3		0x0000	0xFFFF	RO
8	HR_MAC_ADDR_4_5		0x0000	0xFFFF	RO
		Reserved.			
10	HR_USB_STATUS	USB interface status: 0 → Disconnected 1 → Connected	0	1	RO
		Reserved.			
13	HR_NUMBER_OF_ACTIVE_CH	Number of enabled analog channels.	0	6	RO
14	HR_NUMBER_OF_ACTIVE_CHD	Number of enabled digital channels.	0	6	RO
15	HR_RESET_COUNTERS	Reset of digital channel counters. **Note:** Write 1 resets all the digital counters that are configured to be reset by Modbus-TCP and MQTT.	0	1	RW
16	HR_PWR_STATUS	Power supply status: 0 → Powered by the USB interface 1 → Powered by external supply	1	1	RO
17	HR_STATUS_OF_RECORDS	Number of registers pending submission via MQTT protocol.	0	65535	RO
		Reserved.			
20	HR_LAST_CONFIG_YEAR,	Year of the last configuration.	2016	2080	RO
21	HR_LAST_CONFIG_MONTH,	Month of the last configuration.	1	12	RO
22	HR_LAST_CONFIG_DAY,	Day of the last configuration.	1	31	RO
23	HR_LAST_CONFIG_HOUR,	Hour of the last configuration.	0	23	RO
24	HR_LAST_CONFIG_MINUTE,	Minute of the last configuration.	0	59	RO
25	HR_LAST_CONFIG_SECOND	Second of the last configuration.	0	59	RO
26	HR_CURRENT_YEAR	Current year.	2016	2080	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
27	HR_CURRENT_MONTH	Current month.	1	12	RO
28	HR_CURRENT_DAY	Current day.	1	31	RO
29	HR_CURRENT_HOUR	Current hour.	0	23	RO
30	HR_CURRENT_MINUTE	Current minute.	0	59	RO
31	HR_CURRENT_SECOND	Current second.	0	59	RO
		Reserved.			
34	HR_RESET_COUNTER_CHD1	Resets the digital channel counter 1. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
35	HR_RESET_COUNTER_CHD2	Resets the digital channel counter 2. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
36	HR_RESET_COUNTER_CHD3	Resets the digital channel counter 3. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
37	HR_RESET_COUNTER_CHD4	Resets the digital channel counter 4. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
38	HR_RESET_COUNTER_CHD5	Resets the digital channel counter 5. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
39	HR_RESET_COUNTER_CHD6	Resets the digital channel counter 6. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
		Reserved.			
41	HR_DIGITAL_OUT1_VALUE	Digital output status and control: 0 → OFF 1 → ON Allows you to read and write to the output.	0	1	RW
42	HR_DIGITAL_OUT2_VALUE	Digital output status and control: 0 → OFF 1 → ON Allows you to read and write to the output.	0	1	RW
		Reserved.			
45	HR_CHD1_STATUS	Digital channel status: 0 → Not configured 1 → OK 2 → The configuration has an error	0	2	RO
46	HR_CHD1_VALUE_HIGH	Counter value in 32-bit.	0	65535	RO
47	HR_CHD1_VALUE_LOW		0	65535	RO
48	HR_CHD1_TIME_STAMP_LAST_HIGH	Last event timestamp. 32-bit. Unix format.	0x0000	0xFFFF	RO
49	HR_CHD1_TIME_STAMP_LAST_LOW		0x0000	0xFFFF	RO
		Reserved.			
56	HR_CHD2_STATUS	Digital channel status: 0 → Not configured 1 → OK 2 → The configuration has an error	0	2	RO
57	HR_CHD2_VALUE_HIGH	Counter value in 32-bit.	0	65535	RO
58	HR_CHD2_VALUE_LOW		0	65535	RO
59	HR_CHD2_TIME_STAMP_LAST_HIGH	Last event timestamp. 32-bit. Unix format.	0x0000	0xFFFF	RO
60	HR_CHD2_TIME_STAMP_LAST_LOW		0x0000	0xFFFF	RO
		Reserved.			

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
67	HR_CHD3_STATUS	Digital channel status: 0 → Not configured 1 → OK 2 → The configuration has an error	0	2	RO
68	HR_CHD3_VALUE_HIGH	Counter value in 32-bit.	0	65535	RO
69	HR_CHD3_VALUE_LOW		0	65535	RO
70	HR_CHD3_TIME_STAMP_LAST_HIGH	Last event timestamp. 32-bit. Unix format.	0x0000	0xFFFF	RO
71	HR_CHD3_TIME_STAMP_LAST_LOW		0x0000	0xFFFF	RO
		Reserved.			
78	HR_CHD4_STATUS	Digital channel status: 0 → Not configured 1 → OK 2 → The configuration has an error	0	2	RO
79	HR_CHD4_VALUE_HIGH	Counter value in 32-bit.	0	65535	RO
80	HR_CHD4_VALUE_LOW		0	65535	RO
81	HR_CHD4_TIME_STAMP_LAST_HIGH	Last event timestamp. 32-bit. Unix format.	0x0000	0xFFFF	RO
82	HR_CHD4_TIME_STAMP_LAST_LOW		0x0000	0xFFFF	RO
		Reserved.			
89	HR_CHD5_STATUS	Digital channel status: 0 → Not configured 1 → OK 2 → The configuration has an error	0	2	RO
90	HR_CHD5_VALUE_HIGH	Counter value in 32-bit.	0	65535	RO
91	HR_CHD5_VALUE_LOW		0	65535	RO
92	HR_CHD5_TIME_STAMP_LAST_HIGH	Last event timestamp. 32-bit. Unix format.	0x0000	0xFFFF	RO
93	HR_CHD5_TIME_STAMP_LAST_LOW		0x0000	0xFFFF	RO
		Reserved.			
100	HR_CHD6_STATUS	Digital channel status: 0 → Not configured 1 → OK 2 → The configuration has an error	0	2	RO
101	HR_CHD6_VALUE_HIGH	Counter value in 32-bit.	0	65535	RO
102	HR_CHD6_VALUE_LOW		0	65535	RO
103	HR_CHD6_TIME_STAMP_LAST_HIGH	Last event timestamp. 32-bit. Unix format.	0x0000	0xFFFF	RO
104	HR_CHD6_TIME_STAMP_LAST_LOW		0x0000	0xFFFF	RO
		Reserved.			
109	HR_CH1_STATUS	Analog channel 1 status: 0 → Not configured 1 → OK 2 → The configuration has an error	0	2	RO
		Reserved.			
111	HR_CH1_MV_MA_VALUE_H	Value in the unit of measurement (mA or V). Float 32-bit format.	0x0000	0xFFFF	RO
112	HR_CH1_MV_MA_VALUE_L		0x0000	0xFFFF	RO
113	HR_CH1_SENSE_USER_RANGE_H	Value in user range. Float 32-bit format. **Note:** This is the same value as the cloud publication.	0x0000	0xFFFF	RO
114	HR_CH1_SENSE_USER_RANGE_L		0x0000	0xFFFF	RO
120	HR_CH2_STATUS	Analog channel 2 status: 0 → Not configured 1 → OK 2 → The configuration has an error	0	2	RO
		Reserved.			
122	HR_CH2_MV_MA_VALUE_H	Value in the unit of measurement (mA or V). Float 32-bit format.	0x0000	0xFFFF	RO
123	HR_CH2_MV_MA_VALUE_L		0x0000	0xFFFF	RO
124	HR_CH2_SENSE_USER_RANGE_H	Value in user range. Float 32-bit format.	0x0000	0xFFFF	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
125	HR_CH2_SENSE_USER_RANGE_L	**Note:** This is the same value as the cloud publication.	0x0000	0xFFFF	RO
		Reserved.			
130	HR_MQTT_LAST_UPDATE_YEAR	Year of last sending to the MQTT Broker.	1	1	RO
131	HR_MQTT_LAST_UPDATE_MONTH	Month of the last sending to the MQTT Broker.	1	12	RO
132	HR_MQTT_LAST_UPDATE_DAY	Day of the last sending to the MQTT Broker.	1	31	RO
133	HR_MQTT_LAST_UPDATE_HOUR	Time of the last sending to the MQTT Broker.	0	23	RO
134	HR_MQTT_LAST_UPDATE_MINUTE	Minute of the last sending to the MQTT Broker.	0	59	RO
135	HR_MQTT_LAST_UPDATE_SECOND	Second of the last sending to the MQTT Broker.	0	59	RO
136	HR_MQTT_STATUS_BROKER	Communication status with the MQTT Broker: 0 → Broker disconnected 1 → Broker connected 2 → DNS problem 3 → Broker error 4 → Connecting to the Broker	0	4	RO
		Reserved.			
139	HR_WIFI_RSSI	Signal quality between the device and the Wi-Fi Gateway displayed in percent. The higher the value, the better the signal.	0	65535	RO
		Reserved.			
141	HR_LAN_GATEWAY_COM_STATUS	ETH communication status: 0 → Gateway disconnected 1 → Gateway connected 2 → Wi-Fi provisioning error 3 → Obtaining IP via DHCP 4 → Error obtaining IP via DHCP	0	4	RO
142	HR_LAN_IP_ADDR_0_1	IPv4 address. Two octets per register. Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RO
143	HR_LAN_IP_ADDR_2_3		0	65535	RO
144	HR_LAN_MASK_ADDR_0_1	Mask. Two octets per register. Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RO
145	HR_LAN_MASK_ADDR_2_3		0	65535	RO
146	HR_LAN_GATEWAY_ADDR_0_1	Gateway. Two octets per register. Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RO
147	HR_LAN_GATEWAY_ADDR_2_3		0	65535	RO
148	HR_LAN_DNS_ADDR_0_1	DNS server IP. Two octets per register. Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RO
149	HR_LAN_DNS_ADDR_2_3		0	65535	RO
		Reserved.			
151	HR_LAN_IPV6_ADDR_0_1,	IPv6 address – Local. Hexadecimal format. 0_1 : 2_3 : 4_5 : 6_7 : 8_9 : 10_11 : 12_13 : 14_15	0	65535	RO
152	HR_LAN_IPV6_ADDR_2_3,		0	65535	RO
153	HR_LAN_IPV6_ADDR_4_5,		0	65535	RO
154	HR_LAN_IPV6_ADDR_6_7,		0	65535	RO
155	HR_LAN_IPV6_ADDR_8_9,		0	65535	RO
156	HR_LAN_IPV6_ADDR_10_11,		0	65535	RO
157	HR_LAN_IPV6_ADDR_12_13,		0	65535	RO
158	HR_LAN_IPV6_ADDR_14_15,		0	65535	RO
159	HR_LAN_IPV6_GLOBAL_ADDR_0_1,	IPv6 address – Global. Hexadecimal format. 0_1 : 2_3 : 4_5 : 6_7 : 8_9 : 10_11 : 12_13 : 14_15	0	65535	RO
160	HR_LAN_IPV6_GLOBAL_ADDR_2_3,		0	65535	RO
161	HR_LAN_IPV6_GLOBAL_ADDR_4_5,		0	65535	RO
162	HR_LAN_IPV6_GLOBAL_ADDR_6_7,		0	65535	RO
163	HR_LAN_IPV6_GLOBAL_ADDR_8_9,		0	65535	RO
164	HR_LAN_IPV6_GLOBAL_ADDR_10_11,		0	65535	RO
165	HR_LAN_IPV6_GLOBAL_ADDR_12_13,		0	65535	RO
166	HR_LAN_IPV6_GLOBAL_ADDR_14_15,		0	65535	RO
167	HR_CHD1_LEVEL,	Logical level of digital input 1.	0	1	RO
168	HR_CHD2_LEVEL,	Logical level of digital input 2.	0	1	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
169	HR_CHD3_LEVEL,	Logical level of digital input 3.	0	1	RO
170	HR_CHD4_LEVEL,	Logical level of digital input 4.	0	1	RO
171	HR_CHD5_LEVEL,	Logical level of digital input 5.	0	1	RO
172	HR_CHD6_LEVEL,	Logical level of digital input 6.	0	1	RO
173		Reserved.			
174	HR_CHD1_SETVALUE_H	Changes the value of the 32-bit counter for channel 1.	0	65535	RW
175	HR_CHD1_SETVALUE_L		0	65535	RW
176	HR_CHD2_SETVALUE_H	Changes the value of the 32-bit counter for channel 2.	0	65535	RW
177	HR_CHD2_SETVALUE_L		0	65535	RW
178	HR_CHD3_SETVALUE_H	Changes the value of the 32-bit counter for channel 3.	0	65535	RW
179	HR_CHD3_SETVALUE_L		0	65535	RW
180	HR_CHD4_SETVALUE_H	Changes the value of the 32-bit counter for channel 4.	0	65535	RW
181	HR_CHD4_SETVALUE_L,		0	65535	RW
182	HR_CHD5_SETVALUE_H	Changes the value of the 32-bit counter for channel 5.	0	65535	RW
183	HR_CHD5_SETVALUE_L		0	65535	RW
184	HR_CHD6_SETVALUE_H	Changes the value of the 32-bit counter for channel 6.	0	65535	RW
185	HR_CHD6_SETVALUE_L		0	65535	RW
186		Reserved.			
187	HR_SS_COLLECT_RECORD_MAX_ QTTY	Maximum number of downloads supported by memory.	1824	7096	RO
188	HR_SS_COLLECT_LAST_RECORD	Position of the last download added to memory.	0	7096	RO
189	HR_SS_COLLECT_FIRST_RECORD	Position of the first download added to memory.	0	7096	RO
190	HR_SS_COLLECT_REQUESTED_ RECORD	Position of the download requested for reading.	0	7096	RW
191	HR_SS_COLLECT_TIMESTAMP_UNIX_H	Timestamp of the requested download in Unix format.	0	65535	RO
192	HR_SS_COLLECT_TIMESTAMP_UNIX_L		0	65535	RO
193	HR_SS_COLLECT_TIMESTAMP_MS	Timestamp of the requested download in milliseconds.	0	65535	RO
194	HR_SS_COLLECT_CHD_EVENT_ INDEX	When an event occurs in the requested download, returns the index of the digital channel: 0 → No event. It is a periodic log 1 → Event on channel 1 2 → Event on channel 2 3 → Event on channel 3 4 → Event on channel 4 5 → Event on channel 5 6 → Event on channel 6	0	6	RO
195	HR_SS_COLLECT_CHD_EVENT_TYPE	When an event occurs in the requested download, returns the event type: 0 → No event. It is a periodic log 1 → Falling edge event of the digital channel 2 → Rising edge event of the digital channel	0	2	RO
196	HR_SS_COLLECT_CHD1_VALUE_H	Value of digital channel 1 in the requested download.	0	65535	RO
197	HR_SS_COLLECT_CHD1_VALUE_L		0	65535	RO
198	HR_SS_COLLECT_CHD2_VALUE_H	Value of digital channel 2 in the requested download.	0	65535	RO
199	HR_SS_COLLECT_CHD2_VALUE_L		0	65535	RO
200	HR_SS_COLLECT_CHD3_VALUE_H	Value of digital channel 3 in the requested download.	0	65535	RO
201	HR_SS_COLLECT_CHD3_VALUE_L		0	65535	RO
202	HR_SS_COLLECT_CHD4_VALUE_H	Value of digital channel 4 in the requested download.	0	65535	RO
203	HR_SS_COLLECT_CHD4_VALUE_L		0	65535	RO
204	HR_SS_COLLECT_CHD5_VALUE_H	Value of digital channel 5 in the requested download.	0	65535	RO
205	HR_SS_COLLECT_CHD5_VALUE_L		0	65535	RO
206	HR_SS_COLLECT_CHD6_VALUE_H	Value of digital channel 6 in the requested download.	0	65535	RO
207	HR_SS_COLLECT_CHD6_VALUE_L		0	65535	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
208	HR_SS_COLLECT_CH1_SENSE_USER_RANGE_H	Displays the sensor value in the user range of analog channel 1 (in Float).	0	65535	RO
209	HR_SS_COLLECT_CH1_SENSE_USER_RANGE_L		0	65535	RO
210	HR_SS_COLLECT_CH2_SENSE_USER_RANGE_H	Displays the sensor value in the user range of analog channel 2 (in Float).	0	65535	RO
211	HR_SS_COLLECT_CH2_SENSE_USER_RANGE_L		0	65535	RO
		Reserved.			
216	HR_SS_WIFI_RSSI_MIN	Minimum value indicated by the HR_WIFI_RSSI register. You can use the HR_RESET_DIAG_RSSI register to reset the value.	-120	-20	RO
217	HR_SS_WIFI_RSSI_MAX	Maximum value indicated by the HR_WIFI_RSSI register. You can use the HR_RESET_DIAG_RSSI register to reset the value.	-120	-20	RO
218	HR_SS_WIFI_RSSI_AVERAGE	Average value indicated by the HR_WIFI_RSSI register. You can use the HR_RESET_DIAG_RSSI register to reset the value.	-120	-20	RO
		Reserved.			
221	HR_RESET_COUNTER_WDT	Resets the system Watchdog diagnostic counters.	0	1	RW
222	HR_RESET_COUNTER_LOGS	Resets the system logs diagnostic counters.	0	1	RW
223	HR_RESET_DIAG_RSSI	Resets the minimum, maximum, and average signal quality (RSSI) measurement.	0	1	RW
		Reserved.			
227	HR_SS_CHR1_STATUS	Remote channel 1 status. 0 → Disabled channel. 1 → Action not allowed on the slave device. 2 → Register address not allowed. 3 → Query value not allowed. 4, 5, 6, 7, and 8 → Reserved. 9 → Communication timeout. 10 → CRC failure in the device response. 11 → Successful communication.	0	11	RO
228	HR_SS_CH_REMOTE_01_VALUE_HH	Reports the value of the last reading from remote channel 1. The layout in the registers follows the high first and depends on the type of data configured in the channel (whether it is 16 bits, 32 bits, or 64 bits). If the channel is set to 16 bits, the value will be in the final _LL register. If it is set to 32 bits, the data will be in the _LH and _LL registers. If it is set to 64 bits, the value will be in the 4 registers _HH, _HL, _LH, and _LL.	0x0000	0xFFFF	RO
229	HR_SS_CH_REMOTE_01_VALUE_HL		0x0000	0xFFFF	RO
230	HR_SS_CH_REMOTE_01_VALUE_LH		0x0000	0xFFFF	RO
231	HR_SS_CH_REMOTE_01_VALUE_LL		0x0000	0xFFFF	RO
232	HR_SS_CH_REMOTE_01_DOUBLE_HH	Remote channel 1 value in double 64-bit format. The layout of the double 64-bit data in the 4 Modbus registers follows high part first.	0x0000	0xFFFF	RO
233	HR_SS_CH_REMOTE_01_DOUBLE_HL		0x0000	0xFFFF	RO
234	HR_SS_CH_REMOTE_01_DOUBLE_LH		0x0000	0xFFFF	RO
235	HR_SS_CH_REMOTE_01_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
237	HR_SS_CHR2_STATUS	Remote channel 2 status.	0	11	RO
238	HR_SS_CH_REMOTE_02_VALUE_HH	Reports the value of the last reading from remote channel 2.	0x0000	0xFFFF	RO
239	HR_SS_CH_REMOTE_02_VALUE_HL		0x0000	0xFFFF	RO
240	HR_SS_CH_REMOTE_02_VALUE_LH		0x0000	0xFFFF	RO
241	HR_SS_CH_REMOTE_02_VALUE_LL		0x0000	0xFFFF	RO
242	HR_SS_CH_REMOTE_02_DOUBLE_HH	Remote channel 2 value in double 64-bit format.	0x0000	0xFFFF	RO
243	HR_SS_CH_REMOTE_02_DOUBLE_HL		0x0000	0xFFFF	RO
244	HR_SS_CH_REMOTE_02_DOUBLE_LH		0x0000	0xFFFF	RO
245	HR_SS_CH_REMOTE_02_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
247	HR_SS_CHR3_STATUS	Remote channel 3 status.	0	11	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
248	HR_SS_CH_REMOTE_03_VALUE_HH	Reports the value of the last reading from remote channel 3.	0x0000	0xFFFF	RO
249	HR_SS_CH_REMOTE_03_VALUE_HL		0x0000	0xFFFF	RO
250	HR_SS_CH_REMOTE_03_VALUE_LH		0x0000	0xFFFF	RO
251	HR_SS_CH_REMOTE_03_VALUE_LL		0x0000	0xFFFF	RO
252	HR_SS_CH_REMOTE_03_DOUBLE_HH	Remote channel 3 value in double 64-bit format.	0x0000	0xFFFF	RO
253	HR_SS_CH_REMOTE_03_DOUBLE_HL		0x0000	0xFFFF	RO
254	HR_SS_CH_REMOTE_03_DOUBLE_LH		0x0000	0xFFFF	RO
255	HR_SS_CH_REMOTE_03_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
247	HR_SS_CHR4_STATUS	Remote channel 4 status.	0	11	RO
258	HR_SS_CH_REMOTE_04_VALUE_HH	Reports the value of the last reading from remote channel 4.	0x0000	0xFFFF	RO
259	HR_SS_CH_REMOTE_04_VALUE_HL		0x0000	0xFFFF	RO
260	HR_SS_CH_REMOTE_04_VALUE_LH		0x0000	0xFFFF	RO
261	HR_SS_CH_REMOTE_04_VALUE_LL		0x0000	0xFFFF	RO
262	HR_SS_CH_REMOTE_04_DOUBLE_HH	Remote channel 4 value in double 64-bit format.	0x0000	0xFFFF	RO
263	HR_SS_CH_REMOTE_04_DOUBLE_HL		0x0000	0xFFFF	RO
264	HR_SS_CH_REMOTE_04_DOUBLE_LH		0x0000	0xFFFF	RO
265	HR_SS_CH_REMOTE_04_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
267	HR_SS_CHR5_STATUS	Remote channel 5 status.	0	11	RO
268	HR_SS_CH_REMOTE_05_VALUE_HH	Reports the value of the last reading from remote channel 5.	0x0000	0xFFFF	RO
269	HR_SS_CH_REMOTE_05_VALUE_HL		0x0000	0xFFFF	RO
270	HR_SS_CH_REMOTE_05_VALUE_LH		0x0000	0xFFFF	RO
271	HR_SS_CH_REMOTE_05_VALUE_LL		0x0000	0xFFFF	RO
272	HR_SS_CH_REMOTE_05_DOUBLE_HH	Remote channel 5 value in double 64-bit format.	0x0000	0xFFFF	RO
273	HR_SS_CH_REMOTE_05_DOUBLE_HL		0x0000	0xFFFF	RO
274	HR_SS_CH_REMOTE_05_DOUBLE_LH		0x0000	0xFFFF	RO
275	HR_SS_CH_REMOTE_05_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
277	HR_SS_CHR6_STATUS	Remote channel 6 status.	0	11	RO
278	HR_SS_CH_REMOTE_06_VALUE_HH	Reports the value of the last reading from remote channel 6.	0x0000	0xFFFF	RO
279	HR_SS_CH_REMOTE_06_VALUE_HL		0x0000	0xFFFF	RO
280	HR_SS_CH_REMOTE_06_VALUE_LH		0x0000	0xFFFF	RO
281	HR_SS_CH_REMOTE_06_VALUE_LL		0x0000	0xFFFF	RO
282	HR_SS_CH_REMOTE_06_DOUBLE_HH	Remote channel 6 value in double 64-bit format.	0x0000	0xFFFF	RO
283	HR_SS_CH_REMOTE_06_DOUBLE_HL		0x0000	0xFFFF	RO
284	HR_SS_CH_REMOTE_06_DOUBLE_LH		0x0000	0xFFFF	RO
285	HR_SS_CH_REMOTE_06_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
287	HR_SS_CHR7_STATUS	Remote channel 7 status.	0	11	RO
288	HR_SS_CH_REMOTE_07_VALUE_HH	Reports the value of the last reading from remote channel 7.	0x0000	0xFFFF	RO
289	HR_SS_CH_REMOTE_07_VALUE_HL		0x0000	0xFFFF	RO
290	HR_SS_CH_REMOTE_07_VALUE_LH		0x0000	0xFFFF	RO
291	HR_SS_CH_REMOTE_07_VALUE_LL		0x0000	0xFFFF	RO
292	HR_SS_CH_REMOTE_07_DOUBLE_HH	Remote channel 7 value in double 64-bit format.	0x0000	0xFFFF	RO
293	HR_SS_CH_REMOTE_07_DOUBLE_HL		0x0000	0xFFFF	RO
294	HR_SS_CH_REMOTE_07_DOUBLE_LH		0x0000	0xFFFF	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
295	HR_SS_CH_REMOTE_07_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
297	HR_SS_CHR8_STATUS	Remote channel 8 status.	0	11	RO
298	HR_SS_CH_REMOTE_08_VALUE_HH	Reports the value of the last reading from remote channel 8.	0x0000	0xFFFF	RO
299	HR_SS_CH_REMOTE_08_VALUE_HL		0x0000	0xFFFF	RO
300	HR_SS_CH_REMOTE_08_VALUE_LH		0x0000	0xFFFF	RO
301	HR_SS_CH_REMOTE_08_VALUE_LL		0x0000	0xFFFF	RO
302	HR_SS_CH_REMOTE_08_DOUBLE_HH	Remote channel 8 value in double 64-bit format.	0x0000	0xFFFF	RO
303	HR_SS_CH_REMOTE_08_DOUBLE_HL		0x0000	0xFFFF	RO
304	HR_SS_CH_REMOTE_08_DOUBLE_LH		0x0000	0xFFFF	RO
305	HR_SS_CH_REMOTE_08_DOUBLE_LL		0x0000	0xFFFF	RO

Table 10

8 CONFIGURATION SOFTWARE

NXperience software is the main tool to configure and perform the **DigiRail IoT** diagnosis and allows you to explore all the device features, communicating through its USB interface or via Modbus-TCP. However, **NXperience** is not a supervisory system and has no MQTT Broker functionality. You must use appropriate systems for the application to enjoy all the benefits provided by the device.

This manual describes the generic functionalities of the software. For more information, check the specific operations manual. The software can be downloaded free of charge from our website www.novusautomation.com, in the Download Area.

8.1 CONFIGURING DIGIRAIL IOT WITH NXPERIENCE

You can configure **DigiRail IoT** by clicking the **Configure** button, located on the **NXperience** home screen. The following sections describe each of the parameters that can be configured and their particularities.

8.1.1 GENERAL SETTINGS

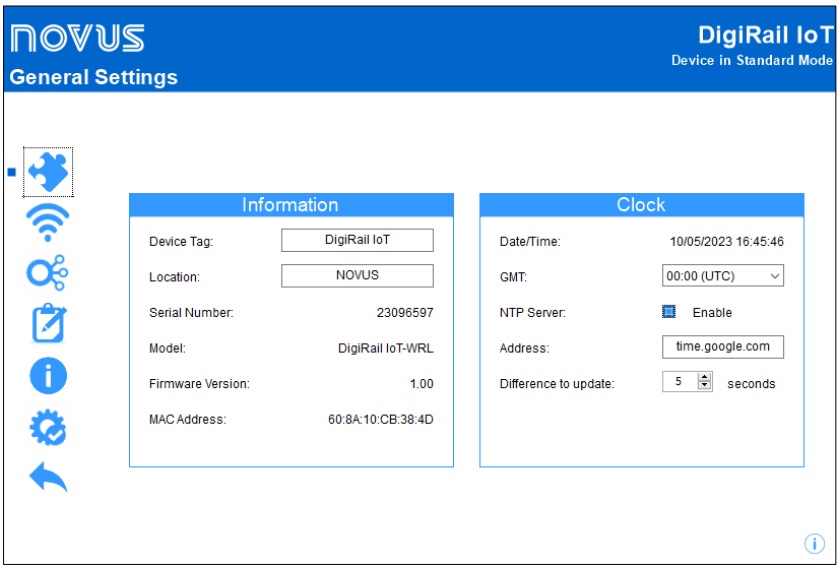

Figure 19

INFORMATION

- **Device Tag:** Allows you to configure a tag for the device. The field allows up to 20 characters.
- **Location:** Allows you to inform the place where the device has been positioned. The field allows up to 40 characters.
- **Serial Number:** Displays the device serial number.
- **Model:** Displays the device model.
- **Firmware Version:** Displays the device firmware version.
- **MAC Address:** Displays the device MAC address.

CLOCK

- **Date/Time**: Displays the date and time of the Windows system, which will be used by **NXperience** to set the device clock when sending the configuration.
- **GMT**: Allows you to configure the GMT of the place where the device will be used (preferably during the first use).
- **NTP Server:** Once enabled, allows you to perform automatic clock synchronization through an NTP server.
  - **Address:** If the "NTP Server" option is enabled, allows you to set an address for the NTP server and thus update the clock automatically.
  - **Difference to update:** If the "NTP Server" option is enabled, the clock will be updated whenever the difference between the clock of the NTP server and the device are greater than the value set in the parameter.

8.1.2 COMMUNICATION

This screen is divided into the following tabs: Ethernet or Wi-Fi, Modbus-TCP Protocol, MQTT Protocol, and RS485.

ETHERNET

This tab is specific to **DigiRail IoT – ETH** model.

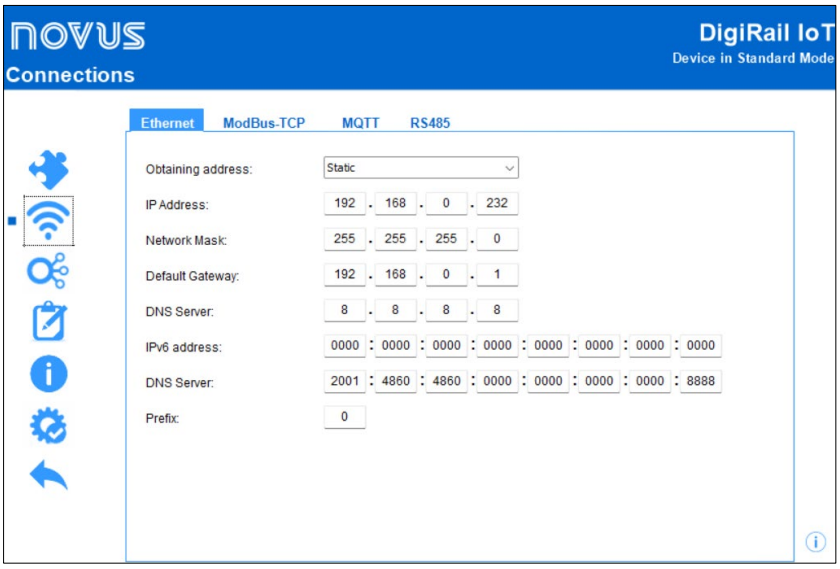

Figure 20

- **Obtaining Address**: Allows you to configure the way in which **DigiRail IoT - ETH** will acquire an IP:
  - **DHCP (Dynamic Host Configuration Protocol):** Protocol that allows the IP address (Internet Protocol) to be assigned by the network server.
  - **Static:** Allows the user to define the IP address, the subnet mask, and the default gateway for the connection. In this case, it is also necessary to define the DNS (Domain Name System) server. By default, the device is configured with the **DHCP** setting.
- **IPv4 Configuration:**
  - **IP Address**: Allows you to configure the IP address. This parameter refers to the identification of the device in a local or public network. Each computer or device on the Internet or in an internal network has a unique IP.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **Network Mask**: Allows you to configure the network mask. This parameter allows you to divide a specific network into smaller subnets, optimizing the use of a certain IP range.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **Default Gateway**: Allows you to define the gateway to be used. This parameter refers to the address that connects the device to the Internet.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **DNS Server**: Allows you to define the DNS server. This parameter refers to a hierarchical and distributed name management system for computers, services or any resource connected to the Internet or to a private network.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
- **IPv6 Configuration:**
  - **IP Address:** Allows you to configure the IPv6 address. This parameter refers to the identification of the device in a local or public network. Each computer or device on the Internet or in an internal network has a unique IP.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **DNS Server:** Allows you to configure the DNS server. This parameter refers to a hierarchical and distributed name management system for computers, services or any resource connected to the Internet or to a private network.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **Prefix:** Allows you to configure the prefix to be used.

WI-FI

This tab is specific to **DigiRail IoT – WRL** model.

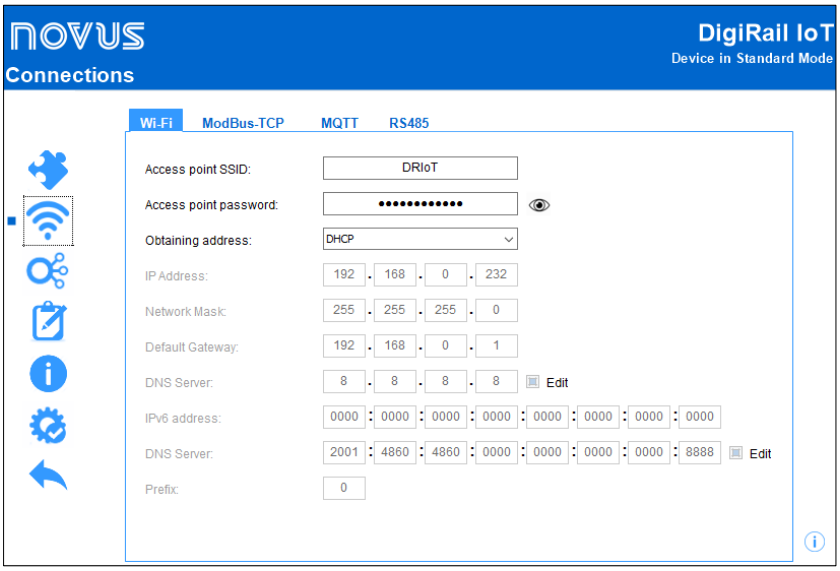

Figure 21

- **Wi-Fi Configuration:**
  - **Access Point SSID**: Allows you to enter the name of the Wi-Fi network to which **DigiRail IoT – WRL** will try to connect to. The field allows up to 32 alphanumeric characters.
  - **Access Point Password**: Allows you to enter the Wi-Fi network password to which **DigiRail IoT – WRL** will try to connect to. The field allows up to 21 alphanumeric characters.

  DigiRail IoT leaves the factory pre-configured to connect to an Access Point with SSID "DRIoT" and password "digirail-iot". If the user has several **DigiRail IoT** to configure, simply configure an Access Point with that SSID and password (or put the smartphone to route the Wi-Fi) so that they all connect. This way, the user will not need to configure each of the devices through the USB interface. A new SSID and password configuration can be done either by USB or by Modbus TCP, through the **NXperience** software, or even through the MQTT protocol.
- **Obtaining Address**: Allows you to configure the way in which **DigiRail IoT – WRL** will acquire an IP: DHCP (Dynamic Host Configuration Protocol), protocol that allows the IP address (Internet Protocol) to be assigned by the network server, or Static, which allows the user to define the IP address, the subnet mask, and the default gateway for the connection. In this case, it is also necessary to define the DNS (Domain Name System) server. By default, the device is configured with the **DHCP** setting.
- **IPv4 Configuration:**
  - **IP Address**: Allows you to configure the IP address. This parameter refers to the identification of the device in a local or public network. Each computer or device on the Internet or in an internal network has a unique IP.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **Network Mask**: Allows you to configure the network mask. This parameter allows you to divide a specific network into smaller subnets, optimizing the use of a certain IP range.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **Default Gateway**: Allows you to define the gateway to be used. This parameter refers to the address that connects the device to the Internet.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **DNS Server**: Allows you to define the DNS server. This parameter refers to a hierarchical and distributed name management system for computers, services or any resource connected to the Internet or to a private network.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
- **IPv6 Configuration:**
  - **IP Address:** Allows you to configure the IPv6 address. This parameter refers to the identification of the device in a local or public network. Each computer or device on the Internet or in an internal network has a unique IP.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **DNS Server:** Allows you to configure the DNS server. This parameter refers to a hierarchical and distributed name management system for computers, services or any resource connected to the Internet or to a private network.

    It is a mandatory field when the **Obtaining Address** parameter is set to **Static**.
  - **Prefix:** Allows you to configure the prefix to be used.

MODBUS-TCP PROTOCOL

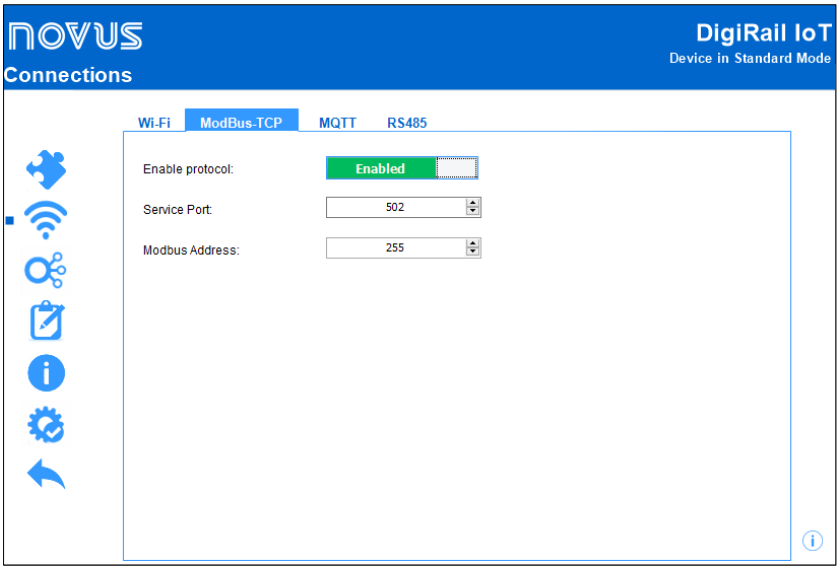

Figure 22

- **Enable Protocol**: Allows you to enable or disable the Modbus-TCP service.
- **Service Port**: Allows you to configure the TCP port on which the service will be available.
- **Modbus Address**: Allows you to configure the Modbus RTU address at which the device will reply as a server (slave). In cases of packets with address that diverge from the configured value, the device will operate as a Gateway.

MQTT PROTOCOL

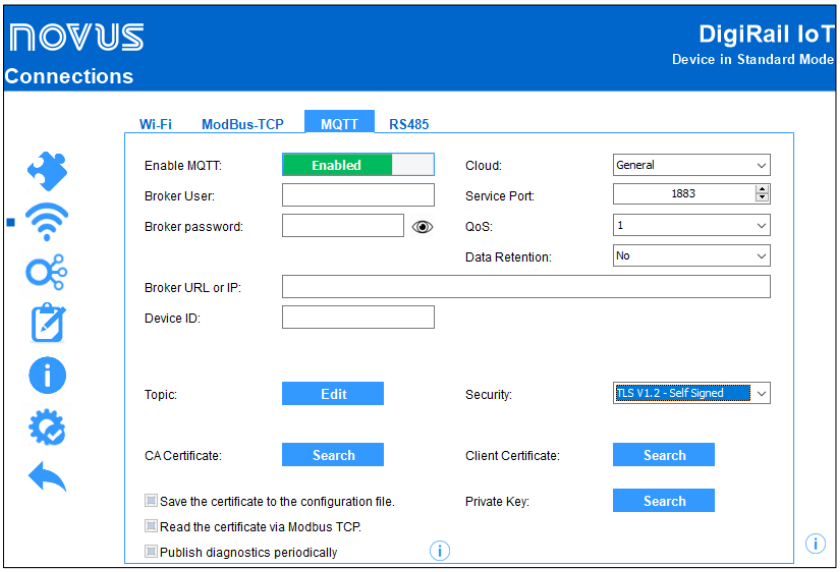

Figure 23

- **Enable MQTT:** Allows you to enable or disable the sending of data via the MQTT protocol.
- **Cloud:** Allows you to configure the platform to be used during the connection with the MQTT Broker: Generic platform, Google Cloud, Amazon AWS, Microsoft Azure, or **NOVUS Cloud**. Depending on the option chosen, the parameters will adjust to meet the specific requirements of the platform. To customize all the parameters, select the **General** option.
- **Broker User:** Allows you to configure the user registered in the Broker. This field allows up to 64 characters. If the field is empty, the connection will be made in anonymous mode. Parameter not necessary for Google Cloud and Microsoft Azure.
- **Broker Password:** Allows you to configure the password of the user registered in the Broker. This field allows up to 64 characters. If the field is empty, the connection will be made in anonymous mode. Parameter not necessary for Google Cloud and Microsoft Azure.
- **Service Port:** Allows you to configure the number of the port used to make the connection with the Broker.
- **QoS:** Allows you to configure the quality level of service used to send MQTT messages: 0 or 1.
- **Data Retention:** Allows you to configure whether data should be retained in the cloud. Not all platforms support this feature.
- **Broker URL or IP:** Allows you to configure the Broker address, which can be either a URL (Uniform Resource Locator) or an IP. The field allows up to 60 characters.
- **Device ID:** Allows you to configure a device ID.
- **Project ID:** Allows you to configure a project ID. Parameter exclusive to Google Cloud.
- **Register ID:** Allows you to configure a register ID. Parameter exclusive to Google Cloud.

- **Region:** Allows you to configure a region for the connection: Us-central1, Europe-west1 or Asia-east1. Parameter exclusive to Google Cloud.
- **Topics:** By clicking the **Edit** button, you can enter the topics to be used for the connection:

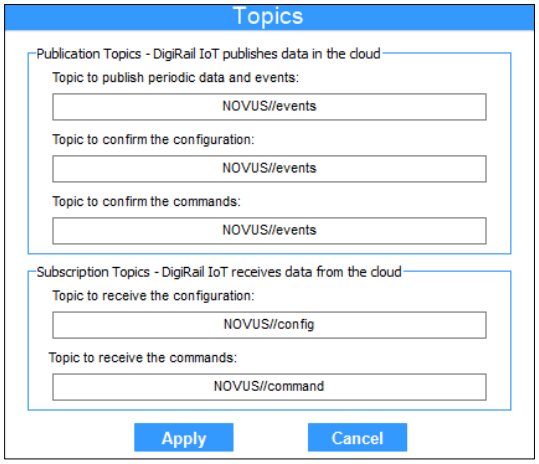

Figure 24

  - **Publishing Topics:** Allows you to configure the device to publish data in the cloud. For more information on publication topics, check the PUBLICATION AND SUBSCRIPTION TOPICS section of the MQTT PROTOCOL chapter.
    - Topic to publish periodic data and events
    - Topic to confirm the configuration
    - Topic to confirm the command
  - **Subscription Topics:** Allows you to configure the device to receive data in the cloud. For more information on subscription topics, check the PUBLICATION AND SUBSCRIPTION TOPICS section of the MQTT PROTOCOL chapter.
    - Topic to receive the configuration
    - Topic to receive the commands
- **Primary Key:** Allows you to configure the primary key to be used. Parameter exclusive to Microsoft Azure.
- **Security:** Allows you to configure the protocol and data encryption for secure communication with the MQTT Broker.
  - **None:** The connection does not use security measures.
  - **Only TLS V 1.2 – CA:** If this option is selected, communication with the Broker will use the Transport Layer Security (TLS) 1.2 protocol, which requires a TLS certificate recognized by a certification authority (CA) to ensure privacy and data integrity.
  - **TLS V 1.2 – Self Signed:** If this option is selected, communication with the Broker will use the Transport Layer Security (TLS) 1.2 protocol, which, in addition to the TLS certificate recognized by a certification authority (CA), also requires authentication of the client certificate and its private key to ensure privacy and data integrity.

    Note: CA certificate, client certificate and private key files are accepted in .pem and .der formats only.
- **Save the certificate to the configuration file:** Once enabled, adds the certificate contents whenever you save a configuration file.
- **Read the certificate via Modbus-TCP:** Once enabled, allows **NXperience** to read certificates via the Modbus-TCP interface.
- **Publish Diagnostics Periodically:** By enabling this parameter, **DigiRail IoT** will perform periodic diagnostic publications in the command confirmation topic. It will be published whenever the device is started and every day at midnight (if connected to the Broker). There will be two publications: One with the "diag" object with the system event counts and another with the "logs" object, returning the last 50 system events.

RS485

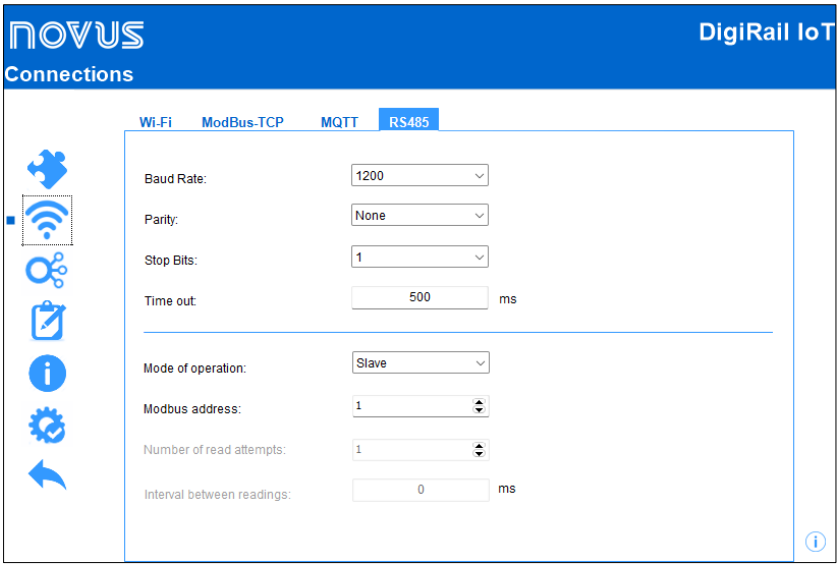

Figure 25

- **Stop Bits:** Allows you to configure the number of Stop Bits to be used by the RS485 interface.
- **Baud Rate:** Allows you to configure the Baud Rate to be used by RS485: 1200, 2400, 4800, 9600, 19200, 38400, 57600 or 115200.
- **Parity:** Allows you to configure the parity to be used by the RS485 interface: Even, odd or none.
- **Timeout:** Allows you to configure a period (in ms) to be used by the RS485 interface to define how long the device will wait for a response from a network slave. This parameter may be configured with a minimum value of 10 ms and a maximum value of 65535 ms.
- **Mode of Operation:** Allows you to configure the type of operation of the RS485 interface: Gateway, master or slave.
- **Modbus Address:** Allows the Modbus-RTU address to be configured when the RS485 interface is operating in slave mode.
- **Number of Read Attempts:** Allows you to configure the number of read attempts the device will make when the RS485 interface is configured in master mode. Parameter with a minimum setting (default setting) of 1 attempt.
- **Interval between Readings:** When in master mode, allows you to set the interval for sending commands between reading attempts.

8.1.3 CHANNELS

ANALOG CHANNELS

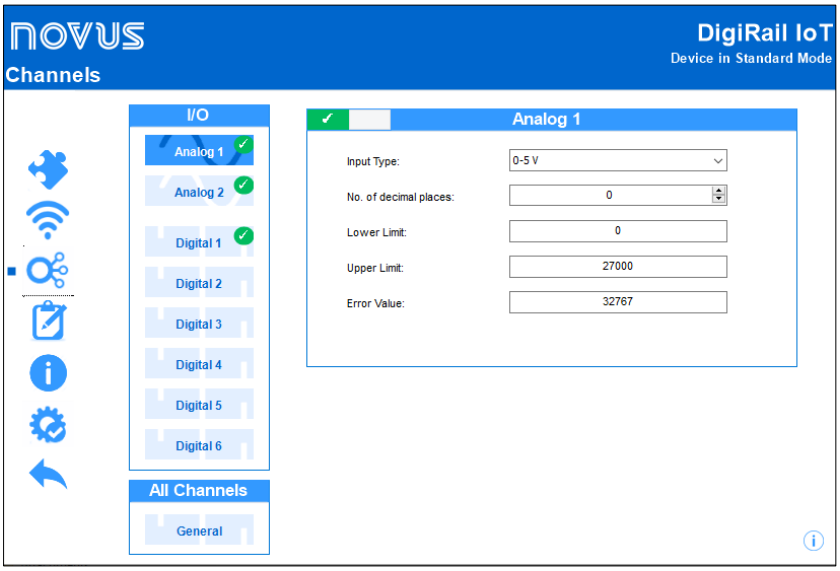

Figure 26

- **Input Type:** Allows you to configure the type of sensor to be used on each analog channel.
- **Number of decimal places:** Allows you to configure the number of decimal places to be used when publishing the calculated value.
- **Lower Limit:** Allows you to configure a minimum value for the sensor.
- **Upper Limit:** Allows you to configure a maximum value for the sensor.
- **Error Value:** Allows you to configure the error value to be considered for the display when an error is detected while reading the sensor.

DIGITAL CHANNELS

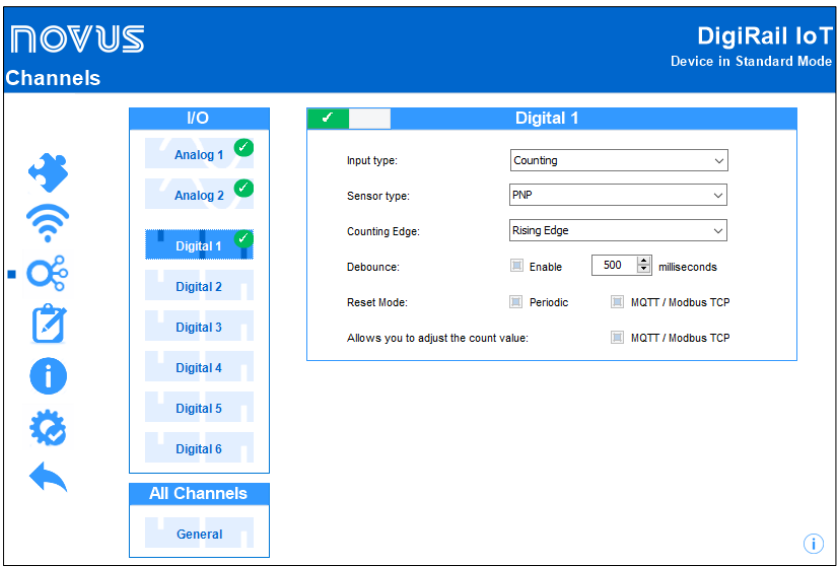

Figure 27

- **Input Type:** Allows you to configure the type of digital input: Counting or Event.
- **Sensor Type:** Allows you to configure the type of sensor to be used: PNP, NPN or Dry Contact.
- **Counting Edge:** Allows you to configure the desired counting edge: Rising edge, falling edge or both edges. This way, the device will increase the counts or recognize an event whenever the configured edge is detected in the digital input.
- **Debounce:** Once enabled, allows you to configure the debounce period to be used. The debounce refers to the sensor settling time (minimum time in which the sensor must remain at the logical level of interest so that the detected edge is considered valid).
- **Reset Mode:** Allows you to configure the reset mode of the selected channel: Periodic and/or MQTT/Modbus TCO. You can set the Periodic mode on the **ALL CHANNELS** tab (see the ALL CHANNELS section of this chapter).
- **Allows you to adjust the count value:** Once enabled, allows changes via Modbus/MQTT in the channel digital counter.

ALL CHANNELS

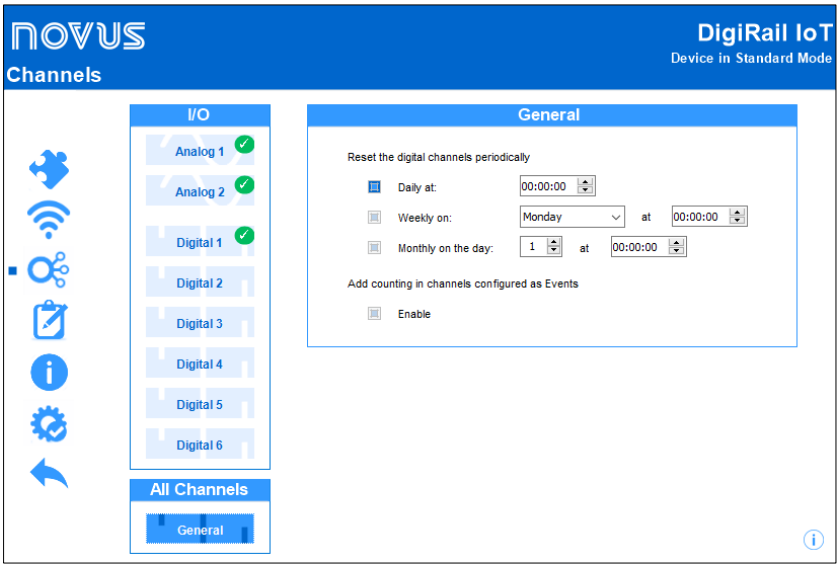

Figure 28

RESET THE DIGITAL CHANNELS PERIODICALLY

It allows you to configure the periodic reset mode of the digital channels configured in **Periodic** mode (see the CHANNELS section of this chapter).

ADD COUNTING IN CHANNELS CONFIGURED AS EVENT

When the digital channel is configured as **Event**, it allows you to add the count value in the circular buffer and in the MQTT publication.

REMOTE CHANNELS

To display the **Remote Channels** screen, you must set the **Mode of Operation** parameter of the RS485 screen to the **Master** option. Otherwise, it will not be displayed.

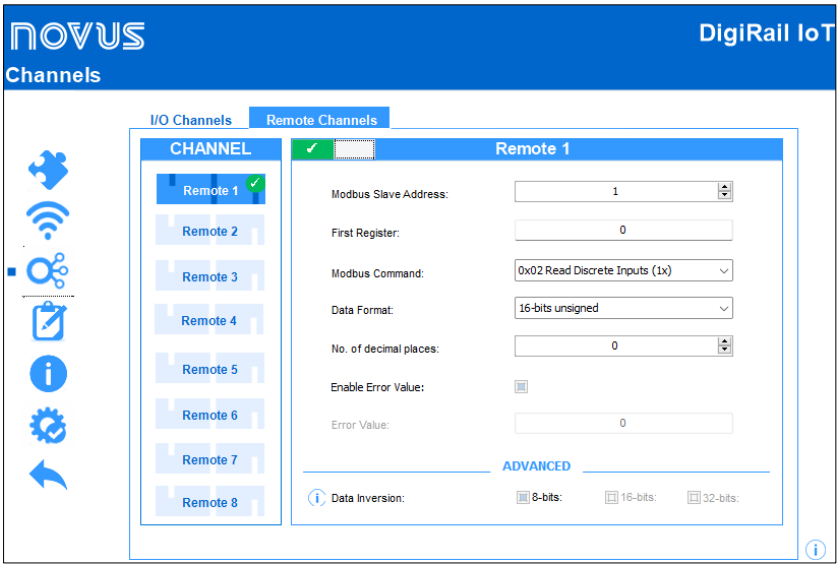

Figure 29

- **Modbus Slave Address:** Allows you to set the Modbus-RTU address of the slave device.
- **First Register:** Allows you to define the initial register from which you want to read the data.
- **Modbus Command:** Allows you to set the Modbus command to be used when sending the read command.
- **Data Format:** Allows you to set the type of data to be read from the address of the initial register.
- **Number of Decimal Places:** Allows you to set the number of decimal places (if the data in the respective register has decimal places).
- **Enable Error Value:** Allows you to enable the display of an error value when read failures occur.
- **Error Value:** Allows you to set the error value for the respective virtual channel.
- **Data Inversion:** Allows you to define an advanced setting for inverting the data read from the register. This setting should be applied whenever it is necessary to adjust the sorting (endianness) of the data bytes.

  Example: By enabling the 8-bit inversion of a hexadecimal value read from slave device 0x1234, it will be registered in **DigiRail IoT** as 0x3412.

8.1.4 LOGS

Figure 30

- **Log Interval:** Allows you to define the interval (in seconds) at which the data will be logged to the circular buffer. If the MQTT protocol is enabled, this interval will also be used to log the periodic data publications.
- **Automatic switch to Alternative Log Interval:** Once enabled, allows increasing the time interval in which the data will be logged in the memory in cases of instability of the connection with the MQTT Broker. When the MQTT publication submission queue is greater than 10% of capacity, the log interval will be changed to the value set in the **Alternative Log Interval** parameter. When the connection is restored and the queue is below 10 % of capacity, the log range is reset.
- **Alternative Log Interval:** Allows you to define the interval (in seconds) to be used when the MQTT publish sending queue is over 10% of capacity. In this case, it the option **Enables automatic switching to Alternative Log** must be enabled.

8.1.5 DIAGRAM

This section shows information about the installation and electrical connections of the device.

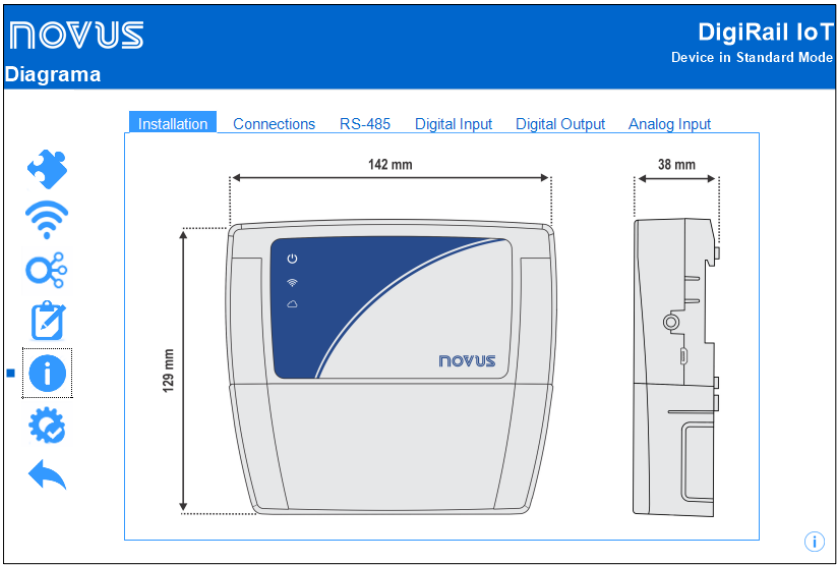

Figure 31

8.2 DIAGNOSTICS

You can view the **DigiRail IoT** diagnosis tab by clicking the **Diagnostics** button located on the **NXperience** home screen.

8.2.1 INFORMATION

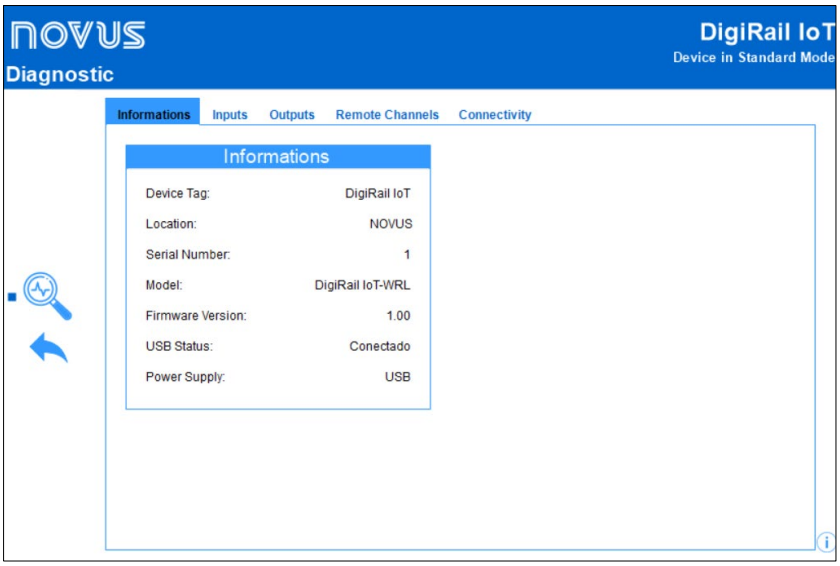

Figure 32

- **Device Tag:** Displays the device tag.
- **Location:** Displays the location of the device, as configured in the General Information section of the Configuration tab (see CONFIGURING DIGIRAIL IOT WITH NXPERIENCE section of this chapter).
- **Serial Number:** Displays the device serial number.
- **Model:** Displays the device model.
- **Firmware Version:** Displays the device current firmware version.
- **USB Status:** Displays the USB interface status of the device.
- **Power Supply:** Displays information about the power supply status of the device.

8.2.2 INPUTS

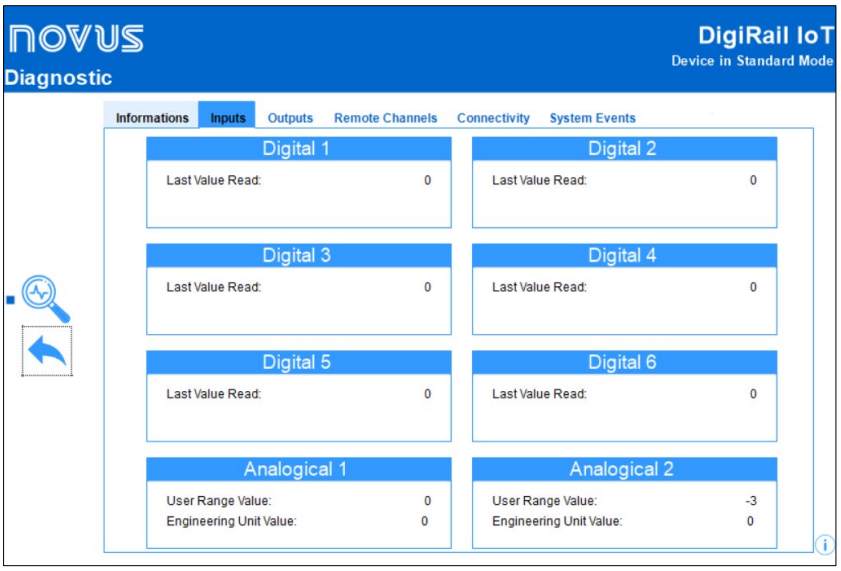

Figure 33

- **Value:** Displays the current value of the configured channel. When the channel has been configured as **Event**, this field will show the value 0 or 1. When the channel has been configured as **Counting**, it will show the counter value.
- **Date/Time:** Displays the date and time of an event if the digital input has been configured in **Event** mode (see the DIGITAL CHANNELS section of this chapter).
- **Engineering Unit Value:** Displays the value measured by the channel in V or mA, depending on the type of channel configured.

8.2.3 OUTPUTS

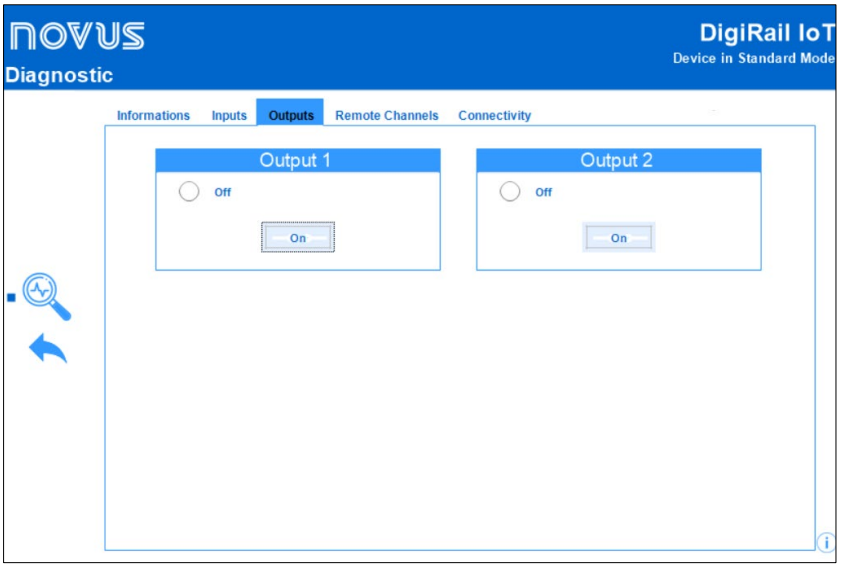

Figure 34

This section allows you to force outputs 1 and 2 in on or off status by clicking the **On** button, in addition to displaying the status of each output.

8.2.4 CONNECTIVITY

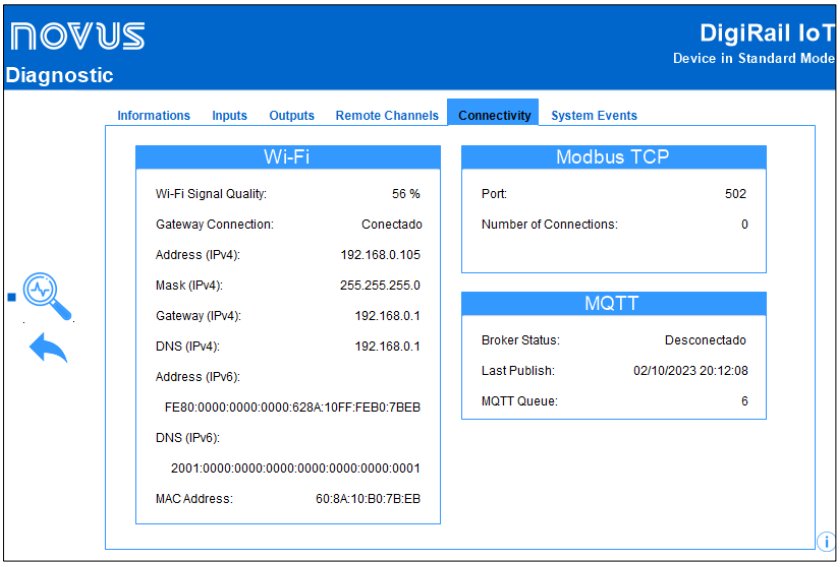

Figure 35

ETHERNET

This section will present parameters related to the device model: **DigiRail IoT – ETH** or **DigiRail IoT – WRL**.

- **Wi-Fi Quality:** Displays the quality of the Wi-Fi signal in percentage value.
- **Gateway Connection:** Displays information on the status of the Gateway connection.
- **IPv4 - Address:** Displays the device IPv4 address.
- **IPv4 - Mask:** Displays the device IPv4 mask.
- **IPv4 - Gateway:** Displays the device Gateway.
- **IPv4 - DNS:** Displays the device DNS.
- **IPv6 - Local:** Displays the device local IPv6 address.
- **IPv6 - Global:** Displays the device global IPv6 address.
- **MAC Address:** Displays the device MAC address.

MODBUS-TCP

- **Port:** Displays the number of the Modbus-TCP port configured in the device.
- **Number of Connections:** Displays the number of Modbus-TCP Clients currently connected to the device.

MQTT

- **Broker Status:** Displays the connection status to the configured MQTT Broker.

- **Last Update:** Displays the day and time of the last package successfully published in the MQTT Broker.
- **MQTT Queue:** Displays the number of logs awaiting publication.

8.2.5 REMOTE CHANNELS

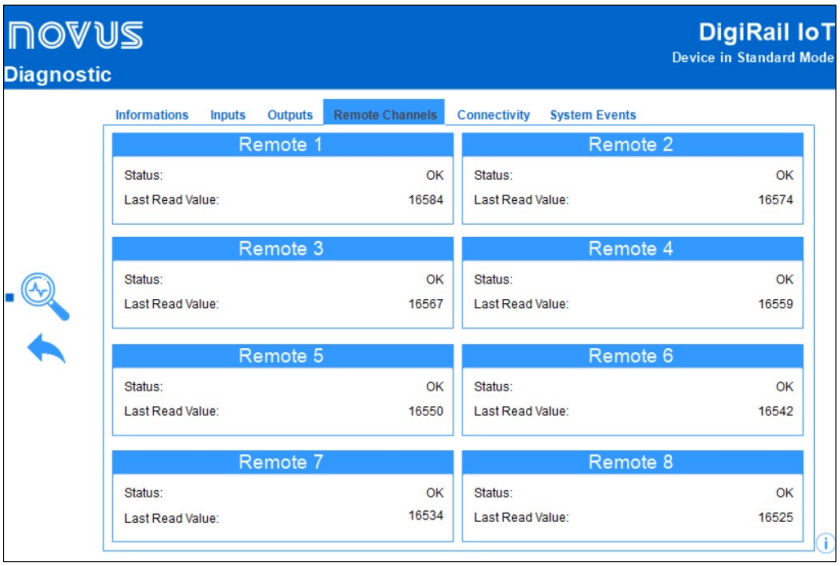

Figure 36 Remote channels

- **Status:** Displays the status of the remote channel. When enabled, a message can be displayed, informing you of the status of the communication (whether it is going as expected or whether there is an error in reading the Modbus register or interpreting the response).
- **Value read:** Displays the value of the last reading made of the register configured for the respective remote channel. If the remote channel and/or communication is incorrect and the error code is enabled and configured, this value will be displayed as the last value obtained.

8.2.6 SYSTEM EVENTS

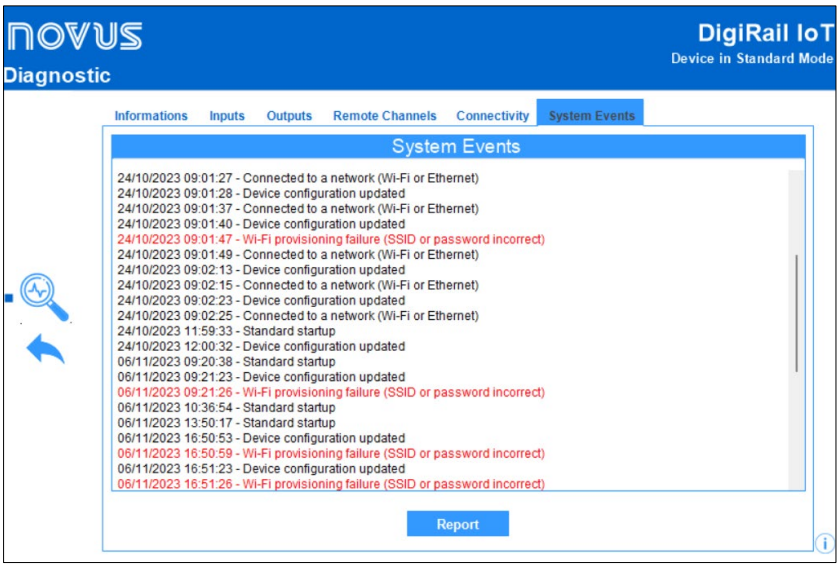

Figure 37

This section allows you to view the system events. In addition, you can generate a report with extension .CSV, which contains the event log and a count of how many times each event occurred.

9 TECHNICAL SPECIFICATION

FEATURES	DIGIRAIL IOT	
Input Channels	6 digital inputs and 2 analog inputs	
Remote Channels	8 configurable channels of up to 64-bits.	
Compatible Analog Signals	0-5 V, 0-10 V, 0-20 mA, 4-20 mA	
Analog Input Resolution	15 bits	
Analog Channels Input Impedance	mA: 15 Ω + 1.5 V V: 1 MΩ	
Accuracy	0.15 % (F.S.)	
Digital Input	**Logical Level**	Logical level "0": < 0.5 V Logical level "1": > 3 V
	Maximum Voltage	30 V
	Input Impedance	5 kΩ
	Input Current @ 30 Vdc (Typical)	7 mA
	Maximum Frequency (Square wave)	Dry Contact: < 10 Hz PNP: 3 kHz NPN: 3 kHz
	Minimum Pulse Length	Dry Contact: 50 ms PNP: 150 us NPN: 150 us
Digital Output	2 digital NPN outputs Maximum current that can be switched at the output: 700 mA	
Buffer Capacity	• 7000 logs with 1 analog input enabled* • 1800 logs with 2 analog inputs enabled and the 6 digital inputs in Count mode*	
Communication Interfaces	**DigiRail IoT – ETH**	• USB 2.0 Interface • Ethernet Interface (10/100 Mbps) with RJ45 connector • RS485 communication interface with Modbus RTU protocol in Gateway mode
	DigiRail IoT – WR	• USB 2.0 Interface • Wi-Fi Interface (802.11 b/g/n 2.4 GHz), supporting WPA-Personal (PSK) WPA/WPA2 TKIP/AES/TKIP and AES encryption • RS485 communication interface with Modbus RTU protocol in Gateway mode
LEDs	• 1 x Status LED • 1 x Local Network Connection LED • 1 x Broker MQTT Connection LED	
Software	**NXperience** (via USB or TCP/IP network for desktops and notebooks).	
Power Supply	**Power Supply**	**Wi-Fi Model:** Consumption: 70 mA @24V Consumption: 160 mA @12V
		Ethernet Model: Consumption: 50 mA @24V Consumption: 120 mA @12V
	Batteries	CR2032 battery for internal clock retention
Dimension	129 mm x 142 mm x 38 mm.	
Mounting	DIN rail or screw mounting.	
Environment	Operating Temperature: -20 to 60 °C (-4 to 140 °F) Storage Temperature: -20 to 60 °C (-4 to 140 °F) Humidity: 5 to 95 % RH (without condensation)	

FEATURES	DIGIRAIL IOT
Housing	ABS+PC
Protection Index	IP20
Certification	ANATEL (09260-20-07089), CE, FCC, Compatible with IEC 60068-2-6 (2007), Contains FCC ID: 2ADHKATWINC1500, Contains IC: 20266-WINC1500PB.

***** None of the cases consider event log.

Table 11

9.1 CIRCULAR BUFFER AVAILABILITY TABLE

This table allows you to evaluate the maximum number of downloads performed by the enabled channels if the digital channel count is in event mode or if it is not in event mode.

DIGITAL CHANNELS	ANALOG CHANNELS	MAXIMUM AMOUNT (WITHOUT EVENT COUNTING)	MAXIMUM AMOUNT (WITH EVENT COUNTING)
0	1	7096	4913
0	2	5806	4913
1	0	5806	4913
2	0	4258	4258
3	0	3361	3361
4	0	2777	2777
5	0	2365	2365
6	0	2060	2060
6	1	1935	1935
6	2	1824	1824

Table 12

REMOTE CHANNELS 16-BITS	REMOTE CHANNELS 32-BITS	REMOTE CHANNELS 64-BITS	MAXIMUM AMOUNT
8	0	0	3041
0	8	0	1726
0	0	8	925

Table 13

For more information about the operation and download of the circular buffer, see the document on the Modbus-TCP Protocol, available on the product page of the **NOVUS** website.

9.2 WIRELESS CONNECTIVITY

DigiRail IoT WRL has an embedded wireless connectivity module to communicate with Wi-Fi 2.4 GHz 802.11 b/g/b networks. The device uses an ATWINC1500-MR210PB Wi-Fi connectivity module from the Microchip manufacturer and allows data connectivity over Wi-Fi.

FEATURES	DESCRIPTION
WLAN standard	IEEE 802 11b/g/n (802.11b/g/n) Part
Frequency ranges	2.412 – 2.484 GHz

Table 14

FEATURES	DESCRIPTION
Frequency ranges	2.412 – 2.484 GHz (2.4 GHz ISM Band)
Number of selectable subchannels	14 Channels
Modulation	802.11b: DBPSK, DQPSK, CCK 802.11g/n: OFDM/64-QAM, 16QAM, QPSK, BPSK
Supported rates	802.11b: 1, 2, 5.5, 11 Mbps 802.11g: 6, 9, 12, 18, 24, 36, 48, 54 Mbps 802.11n (20 MHz, normal GI, 800 ns): 6.5, 13, 19.5, 26, 39, 52, 58.5, 65 Mbps 802.11n (20 MHz, short GI, 400 ns): 7.2, 14.4, 21.7, 28.9, 43.3, 57.8, 65. 72.2 Mbps
Reception: Maximum sensibility	802.11b: -95 dBm @ 1 Mbps, -86 dBm @ 11 Mbps 802.11g: -90 dBm @ 6 Mbps, -74 dBm @ 54 Mbps 802.11n: -89 dBm @ MCS 0, -70.5 dBm @ MCS 7
Output: Maximum power	802.11b: 13.6 dBm @ 1 Mbps, 15.3 dBm @ 11 Mbps 802.11g: 18.9 dBm @ 6 Mbps, 14.3 dBm @ 54 Mbps 802.11n: 18.9 dBm @ MCS 0, 12.2 dBm @ MCS 7

Table 15

9.3 CERTIFICATION

ANATEL

This device is homologated by ANATEL, according to the regulated procedures for conformity assessment of telecommunications devices, and meets the technical requirements applied.

This equipment is not subject to the protection from harmful interference and may not cause interference with duly authorized systems.

For more information, see the ANATEL website: www.gov.br/anatel.

FCC

Contains FCC ID: 2ADHKATWINC1500

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications.

Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate this equipment.

RF Exposure: To satisfy FCC RF exposure requirements, a separation distance of 6.5 cm or more should be maintained between the antenna of this device and persons during operation. To ensure compliance, operations at closer distances than this are not recommended. This device and its antenna(s) must not be co-located or operating in conjunction with any other antenna or transmitter.

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

CE Mark

This is a Class A product. In a domestic environment, this product may cause radio interference in which case the user may be required to take adequate measures.

IC

Contains IC: 20266-WINC1500PB

This device complies with Industry Canada's license exempt RSS standard(s). Operation is subject to the following two conditions:

(1) This device may not cause interference, and

(2) This device must accept any interference, including interference that may cause undesired operation of the device.

The installation of the transmitter must ensure that the antenna has a separation distance of at least 6.5 cm from all persons or compliance must be demonstrated according to the ISED SAR procedure.

VIBRATION TESTS

The device is in accordance with the vibration tests in the profile described in IEC 60068-2-6 (2007) - Environmental Testing - Part 2: Tests - Test Fc: Vibration (Sinusoidal).

10 WARRANTY

Warranty conditions are available on our website www.novusautomation.com/warranty.

11 APPENDIX 1 – RECOMMENDATIONS FOR INSTALLATION IN INDUSTRIAL ENVIRONMENTS

11.1 PURPOSE

Due to the high levels of electromagnetic noise caused by machinery in industrial environments, digital devices can be susceptible to electromagnetic interference. Because of this, good practices should be adopted during the installation of electronic devices to mitigate the effects of this interference.

This appendix presents some recommendations for the installation of digital sensors and is intended to prevent data acquisition problems.

11.2 BEST PRACTICES FOR INDUSTRIAL INSTALLATION

A proper installation must have an industrial grounding system that complies with the IEC 60364-1. This is necessary to ensure the reduction of interference caused by industrial machinery and the equipotentialization between the supply voltages of electronic devices.

Along with the grounding system, it is recommended to choose a good DC 24 V power supply, which guarantees isolation and interference filtering from the AC power input to the DC 24 V power output. CE Mark certified power supplies are the most suitable.

Some manufacturing plants have machines that produce excessive electromagnetic interference. For these cases, it is recommended to choose an instrumentation panel where the electronic devices can be installed. It must comply with the technical standards and provide the shielding of the industrial environment through a grounding terminal that must be connected to the grounding system.

An important recommendation for the proper functioning of the system is to ensure that the cabling between sensors and instrumentation devices has the best path in the manufacturing plant to obtain the shortest distance between instruments and sensors and, at the same time, distance them from possible sources of electromagnetic interference (machines, motors, and sources of electromagnetic pulses). It is recommended that instrumentation sensors run through the plant through grounded conduits exclusively for instrumentation. The power supply network of the machines must run through the plant in separate conduits.

11.3 INSTALLATION RECOMMENDATIONS FOR DIGIRAIL IOT DIGITAL INPUT SIGNALS

In most cases, following good industrial installation practices, described in the previous section, is enough to ensure that the system will work properly. However, depending on the environment where the device is installed, some extra recommendations may be necessary.

11.3.1 ISOLATED GROUND POWER SUPPLY

The figure below illustrates how to connect a power supply to **DigiRail IoT**, a Dry Contact type sensor on digital channel 1, an NPN type sensor on digital channel 2 and a PNP type sensor on digital channel 3. In this example we also show that the power supply must be grounded.

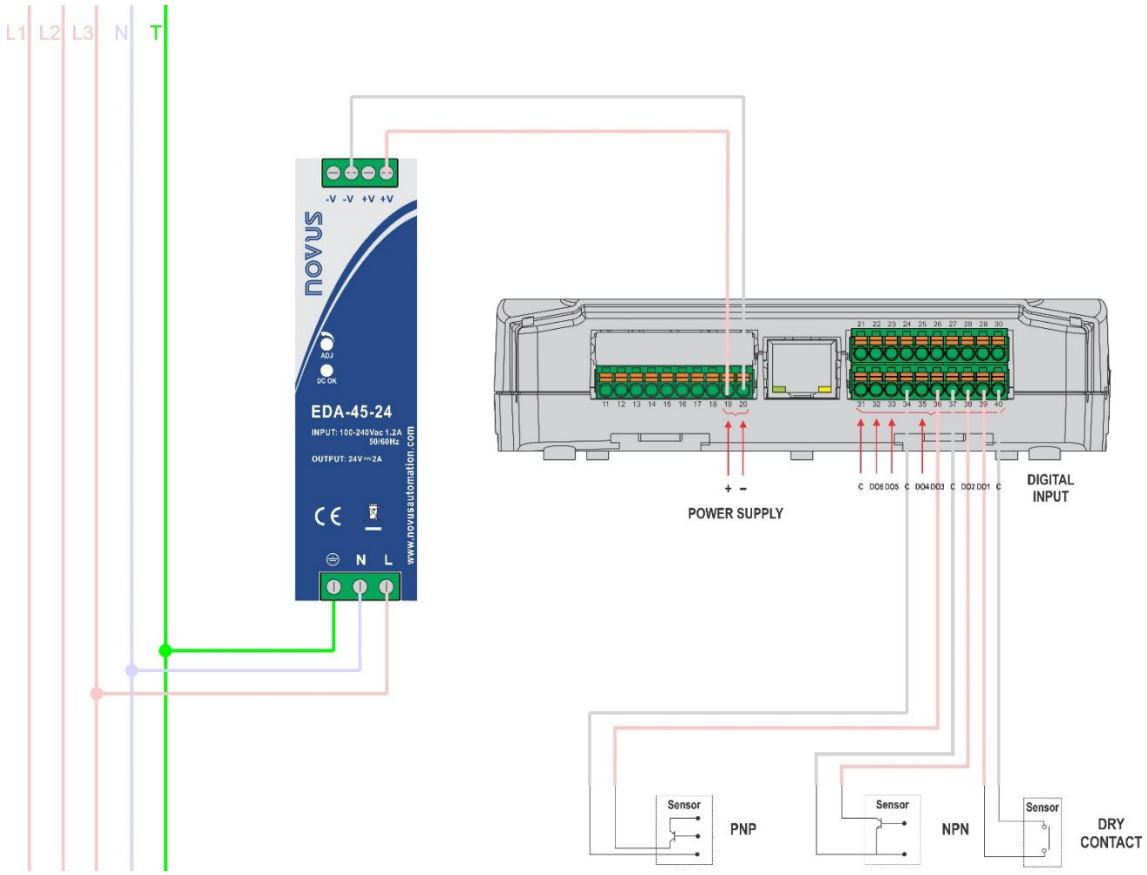

Figure 38

11.3.2 PULL-UP RESISTORS FOR THE SENSORS

If the previous recommendation has been implemented and there is still a problem reading the sensors, pull-up resistors can be used to improve the sensor signal. Dry Contact and NPN sensors should have the sensor reading signal connected to the positive side of the power supply through a 10 kohm ¼ W resistor.

PNP type sensors should have the sensor reading signal connected to the negative of the power supply through a 10 kohm ¼ W resistor (a pull-down resistor). This procedure is used to improve the sensor signal when the sensor is open.

The pull-up and pull-down resistors can be connected either close to the device or close to the sensors. The figure below shows how to connect each of these sensors.

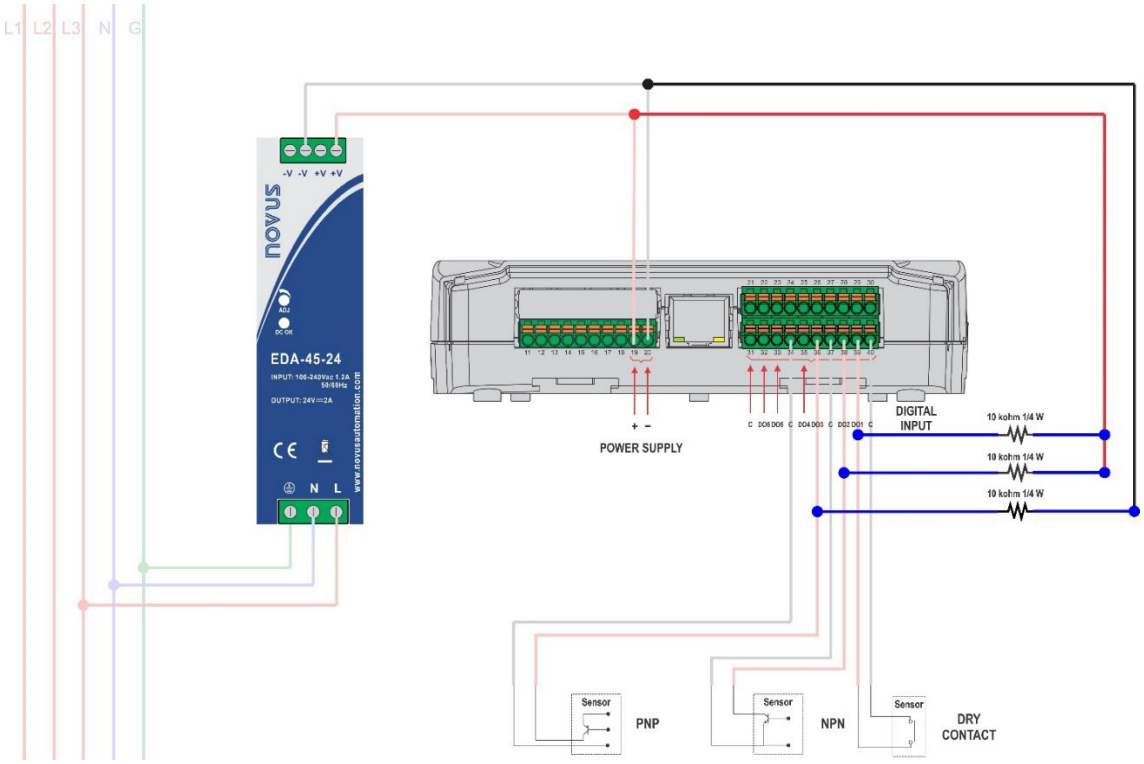

Figure 39

11.3.3 HOW TO GROUND THE NEGATIVE TERMINAL OF THE POWER SUPPLY

If none of the previous implementations has solved the problem, it is possible that there is a potential difference between the negative terminal of the power supply and the system ground, and a current leakage in one of the connected sensors. You can ground the negative terminal of the power supply to eliminate these problems.

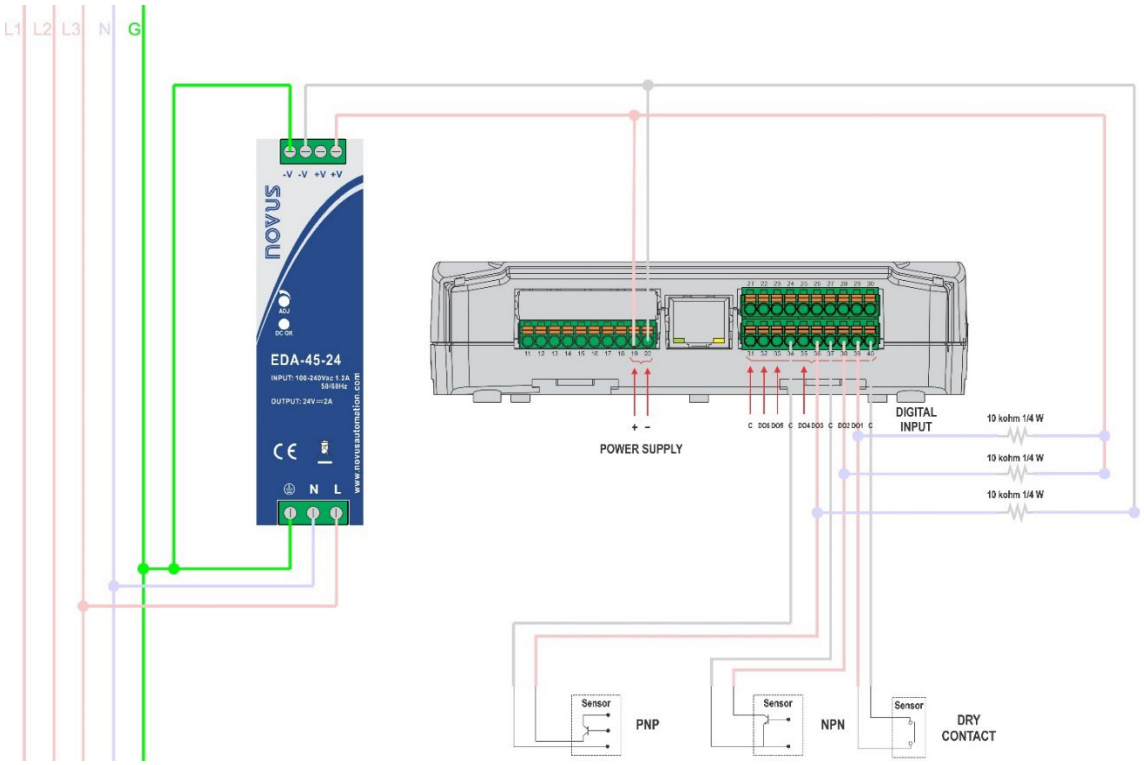

Figure 40

11.3.4 GROUNDED CONDUIT

A good installation practice that prevents possible problems in reading the sensors is to use grounded conduit between the device and sensors. The figure below shows how to use grounded conduit in the path where the sensor signals travel through the plant.

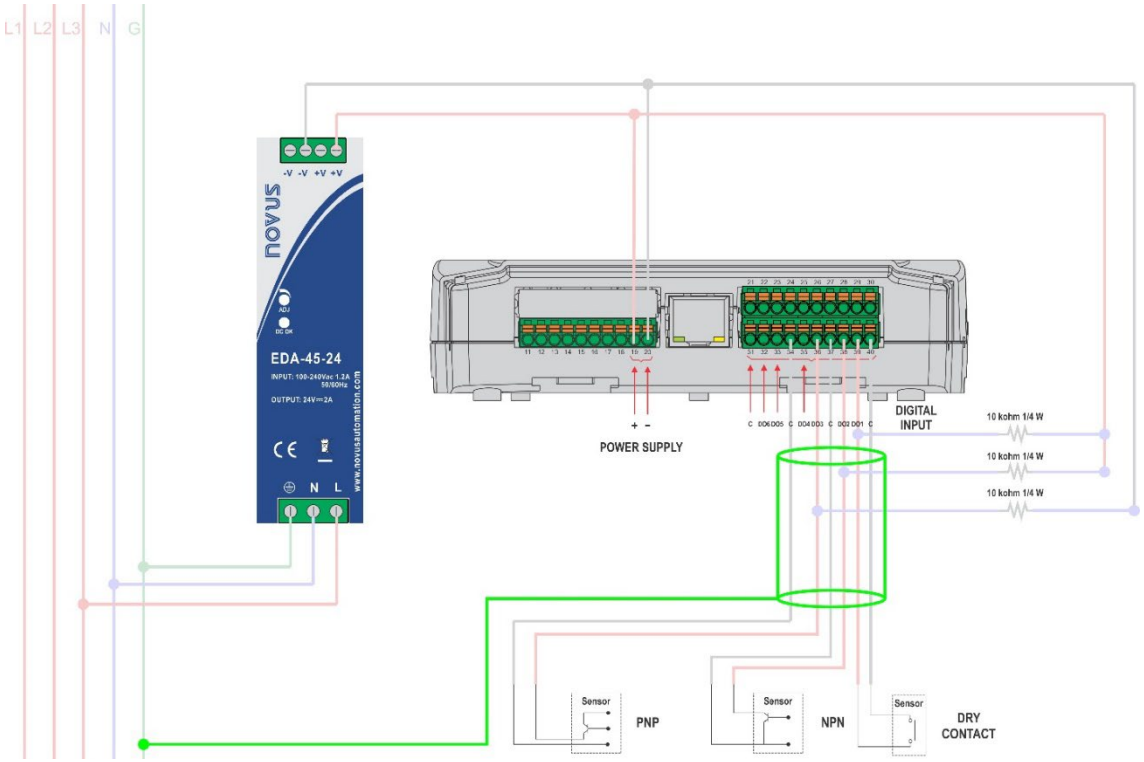

Figure 41

12 APPENDIX 2 – MODBUS-TCP PROTOCOL

12.1 INTRODUCTION

DigiRail IoT is compatible with the Modbus-TCP protocol, a data communication protocol used to connect the device to supervisory control and data acquisition systems (SCADA). It supports up to 3 simultaneous connections and allows up to 3 Modbus-TCP clients (masters) to monitor it at the same time. **DigiRail IoT** operates as a Modbus-TCP server (slave) and as a TCP/RTU Gateway.

As a server (slave), it responds to the configured Modbus RTU address. When configured to operate in Gateway mode, for addresses that differ from the configured address value, it forwards the packet to the RS485 interface and, if there is a reply from a Modbus RTU slave, it replies to the Modbus-TCP client (master) that generated the request.

12.2 REGISTERS

12.1.1 COMMANDS

DigiRail IoT supports the following commands:

READ HOLDING REGISTERS – 0x03:

This command can be used to read the value of one or up to a maximum of 125 consecutive registers, as shown in the table below.

WRITE HOLDING REGISTERS – 0x06:

This command can be used to write to a register, as shown in the table below.

WRITE MULTIPLE HOLDING REGISTERS – 0x16:

This command can be used to write to multiple registers, as shown in the table below.

12.1.2 TABLE OF REGISTERS: STATUS

DigiRail IoT supports the following status registers:

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1	HR_PRODUCT_CODE	Product code.	786	786	RO
2	HR_SERIAL_NUMBER_H	Serial number (32-bits).	0x0000	0xFFFF	RO
3	HR_SERIAL_NUMBER_L		0x0000	0xFFFF	RO
4	HR_FIRMWARE_VERSION	Firmware version x100.	100	65535	RO
		Reserved.			
6	HR_MAC_ADDR_0_1	MAC Address. Hexadecimal format with 2 digits per register. 0 : 1 : 2 : 3 : 4 : 5	0x0000	0xFFFF	RO
7	HR_MAC_ADDR_2_3		0x0000	0xFFFF	RO
8	HR_MAC_ADDR_4_5		0x0000	0xFFFF	RO
		Reserved.			
10	HR_USB_STATUS	USB interface status: 0 → Disconnected. 1 → Connected.	0	1	RO
		Reserved.			
13	HR_NUMBER_OF_ACTIVE_CH	Number of enabled analog channels.	0	6	RO
14	HR_NUMBER_OF_ACTIVE_CHD	Number of enabled digital channels.	0	6	RO
15	HR_RESET_COUNTERS	Reset of digital channel counters. **Note:** Write 1 resets all the digital counters that are configured to be reset by Modbus-TCP and MQTT.	0	1	RW
16	HR_PWR_STATUS	Power supply status: 0 → Powered by the USB interface. 1 → Powered by external supply.	1	1	RO
17	HR_STATUS_OF_RECORDS	Number of registers pending submission via MQTT protocol.	0	65535	RO
		Reserved.			
20	HR_LAST_CONFIG_YEAR,	Year of the last configuration.	2016	2080	RO
21	HR_LAST_CONFIG_MONTH,	Month of the last configuration.	1	12	RO
22	HR_LAST_CONFIG_DAY,	Day of the last configuration.	1	31	RO
23	HR_LAST_CONFIG_HOUR,	Hour of the last configuration.	0	23	RO
24	HR_LAST_CONFIG_MINUTE,	Minute of the last configuration.	0	59	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
25	HR_LAST_CONFIG_SECOND	Second of the last configuration.	0	59	RO
26	HR_CURRENT_YEAR	Current year.	2016	2080	RO
27	HR_CURRENT_MONTH	Current month.	1	12	RO
28	HR_CURRENT_DAY	Current day.	1	31	RO
29	HR_CURRENT_HOUR	Current hour.	0	23	RO
30	HR_CURRENT_MINUTE	Current minute.	0	59	RO
31	HR_CURRENT_SECOND	Current second.	0	59	RO
		Reserved.			
34	HR_RESET_COUNTER_CHD1	Resets the digital channel counter 1. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
35	HR_RESET_COUNTER_CHD2	Resets the digital channel counter 2. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
36	HR_RESET_COUNTER_CHD3	Resets the digital channel counter 3. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
37	HR_RESET_COUNTER_CHD4	Resets the digital channel counter 4. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
38	HR_RESET_COUNTER_CHD5	Resets the digital channel counter 5. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
39	HR_RESET_COUNTER_CHD6	Resets the digital channel counter 6. **Note:** Write 1 resets the counter for this channel if it is configured to allow reset via Modbus-TCP and MQTT protocols.	0	1	RW
		Reserved.			
41	HR_DIGITAL_OUT1_VALUE	Digital output status and control: 0 → Off. 1 → On. Allows you to read and write to the output.	0	1	RW
42	HR_DIGITAL_OUT2_VALUE	Digital output status and control: 0 → Off. 1 → On. Allows you to read and write to the output.	0	1	RW
		Reserved.			
45	HR_CHD1_STATUS	Digital channel status: 0 → Not configured. 1 → OK 2 → The configuration has an error.	0	2	RO
46	HR_CHD1_VALUE_HIGH	Counting mode: Counter value in 32-bit. Event mode: Logical input level.	0	65535	RO
47	HR_CHD1_VALUE_LOW		0	65535	RO
48	HR_CHD1_TIME_STAMP_LAST_ HIGH	Last event timestamp 32-bits. Unix format.	0x0000	0xFFFF	RO
49	HR_CHD1_TIME_STAMP_LAST_ LOW		0x0000	0xFFFF	RO
		Reserved.			
56	HR_CHD2_STATUS	Digital channel status: 0 → Not configured. 1 → OK. 2 → The configuration has an error.	0	2	RO
57	HR_CHD2_VALUE_HIGH	Counting mode: Counter value in 32-bit. Event mode: Logical input level.	0	65535	RO
58	HR_CHD2_VALUE_LOW		0	65535	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
59	HR_CHD2_TIME_STAMP_LAST_ HIGH	Last event timestamp 32-bits. Unix format.	0x0000	0xFFFF	RO
60	HR_CHD2_TIME_STAMP_LAST_ LOW		0x0000	0xFFFF	RO
		Reserved.			
67	HR_CHD3_STATUS	Digital channel status: 0 → Not configured. 1 → OK. 2 → The configuration has an error.	0	2	RO
68	HR_CHD3_VALUE_HIGH	Counting mode: Counter value in 32-bit. Event mode: Logical input level.	0	65535	RO
69	HR_CHD3_VALUE_LOW		0	65535	RO
70	HR_CHD3_TIME_STAMP_LAST_ HIGH	Last event timestamp 32-bits. Unix format.	0x0000	0xFFFF	RO
71	HR_CHD3_TIME_STAMP_LAST_ LOW		0x0000	0xFFFF	RO
		Reserved.			
78	HR_CHD4_STATUS	Digital channel status: 0 → Not configured. 1 → OK. 2 → The configuration has an error.	0	2	RO
79	HR_CHD4_VALUE_HIGH	Counting mode: Counter value in 32-bit. Event mode: Logical input level.	0	65535	RO
80	HR_CHD4_VALUE_LOW		0	65535	RO
81	HR_CHD4_TIME_STAMP_LAST_ HIGH	Last event timestamp 32-bits. Unix format.	0x0000	0xFFFF	RO
82	HR_CHD4_TIME_STAMP_LAST_ LOW		0x0000	0xFFFF	RO
		Reserved.			
89	HR_CHD5_STATUS	Digital channel status: 0 → Not configured. 1 → OK. 2 → The configuration has an error.	0	2	RO
90	HR_CHD5_VALUE_HIGH	Counting mode: Counter value in 32-bit. Event mode: Logical input level.	0	65535	RO
91	HR_CHD5_VALUE_LOW		0	65535	RO
92	HR_CHD5_TIME_STAMP_LAST_ HIGH	Last event timestamp 32-bits. Unix format.	0x0000	0xFFFF	RO
93	HR_CHD5_TIME_STAMP_LAST_ LOW		0x0000	0xFFFF	RO
		Reserved.			
100	HR_CHD6_STATUS	Digital channel status: 0 → Not configured. 1 → OK. 2 → The configuration has an error.	0	2	RO
101	HR_CHD6_VALUE_HIGH	Counting mode: Counter value in 32-bit. Event mode: Logical input level.	0	65535	RO
102	HR_CHD6_VALUE_LOW		0	65535	RO
103	HR_CHD6_TIME_STAMP_LAST_ HIGH	Last event timestamp 32-bits. Unix format.	0x0000	0xFFFF	RO
104	HR_CHD6_TIME_STAMP_LAST_ LOW		0x0000	0xFFFF	RO
		Reserved.			
109	HR_CH1_STATUS	Analog channel 1 status: 0 → Not configured. 1 → OK. 2 → The configuration has an error.	0	2	RO
		Reserved.			
111	HR_CH1_MV_MA_VALUE_H	Value in unit of measurement (mA or V). Float 32-bits format.	0x0000	0xFFFF	RO
112	HR_CH1_MV_MA_VALUE_L		0x0000	0xFFFF	RO
113	HR_CH1_SENSE_USER_RANGE_H	User range value. Float 32-bits format. **Note:** This is the same value as the cloud publication.	0x0000	0xFFFF	RO
114	HR_CH1_SENSE_USER_RANGE_L		0x0000	0xFFFF	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
		Reserved.			
120	HR_CH2_STATUS	Analog channel 2 status: 0 → Not configured. 1 → OK. 2 → The configuration has an error.	0	2	RO
		Reserved.			
122	HR_CH2_MV_MA_VALUE_H	Value in unit of measurement (mA or V). Float 32-bits format.	0x0000	0xFFFF	RO
123	HR_CH2_MV_MA_VALUE_L		0x0000	0xFFFF	RO
124	HR_CH2_SENSE_USER_RANGE_H	User range value. Float 32-bits format. **Note:** This is the same value as the cloud publication.	0x0000	0xFFFF	RO
125	HR_CH2_SENSE_USER_RANGE_L		0x0000	0xFFFF	RO
		Reserved.			
130	HR_MQTT_LAST_UPDATE_YEAR	Year of the last submission to the MQTT Broker.	1	1	RO
131	HR_MQTT_LAST_UPDATE_ MONTH	Month of the last submission to the MQTT Broker.	1	12	RO
132	HR_MQTT_LAST_UPDATE_DAY	Day of the last submission to the MQTT Broker.	1	31	RO
133	HR_MQTT_LAST_UPDATE_HOUR	Hour of the last submission to the MQTT Broker.	0	23	RO
134	HR_MQTT_LAST_UPDATE_ MINUTE	Minute of the last submission to the MQTT Broker.	0	59	RO
135	HR_MQTT_LAST_UPDATE_ SECOND	Second of the last submission to the MQTT Broker.	0	59	RO
136	HR_MQTT_STATUS_BROKER	Communication status with the MQTT Broker: 0 → Disconnected Broker. 1 → Connected Broker. 2 → DNS problem. 3 → Broker error. 4 → Connecting to the Broker.	0	4	RO
		Reserved.			
139	HR_WIFI_RSSI	Signal quality between the device and the Wi-Fi Gateway displayed in percentage. The higher the value, the better the signal.	0	65535	RO
		Reserved.			
141	HR_LAN_GATEWAY_COM_ STATUS	ETH communication status: 0 → Disconnected Gateway. 1 → Connected Gateway. 2 → Wi-Fi provisioning error. 3 → Obtaining IP via DHCP. 4 → Error obtaining IP via DHCP.	0	4	RO
142	HR_LAN_IP_ADDR_0_1	IPv4 address. Two octets per register. Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RO
143	HR_LAN_IP_ADDR_2_3		0	65535	RO
144	HR_LAN_MASK_ADDR_0_1	Mask. Two octets per register. Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RO
145	HR_LAN_MASK_ADDR_2_3		0	65535	RO
146	HR_LAN_GATEWAY_ADDR_0_1	Gateway. Two octets per register. Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RO
147	HR_LAN_GATEWAY_ADDR_2_3		0	65535	RO
148	HR_LAN_DNS_ADDR_0_1	DNS server IP. Two octets per register. Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RO
149	HR_LAN_DNS_ADDR_2_3		0	65535	RO
		Reserved.			
151	HR_LAN_IPV6_ADDR_0_1,	IPv6 address – Local. Hexadecimal format. 0_1 : 2_3 : 4_5 : 6_7 : 8_9 : 10_11 : 12_13 : 14_15	0	65535	RO
152	HR_LAN_IPV6_ADDR_2_3,		0	65535	RO
153	HR_LAN_IPV6_ADDR_4_5,		0	65535	RO
154	HR_LAN_IPV6_ADDR_6_7,		0	65535	RO
155	HR_LAN_IPV6_ADDR_8_9,		0	65535	RO
156	HR_LAN_IPV6_ADDR_10_11,		0	65535	RO
157	HR_LAN_IPV6_ADDR_12_13,		0	65535	RO
158	HR_LAN_IPV6_ADDR_14_15,		0	65535	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
159	HR_LAN_IPV6_GLOBAL_ADDR_0_1,	IPv6 address – Global. Hexadecimal format. 0_1 : 2_3 : 4_5 : 6_7 : 8_9 : 10_11 : 12_13 : 14_15	0	65535	RO
160	HR_LAN_IPV6_GLOBAL_ADDR_2_3,		0	65535	RO
161	HR_LAN_IPV6_GLOBAL_ADDR_4_5,		0	65535	RO
162	HR_LAN_IPV6_GLOBAL_ADDR_6_7,		0	65535	RO
163	HR_LAN_IPV6_GLOBAL_ADDR_8_9,		0	65535	RO
164	HR_LAN_IPV6_GLOBAL_ADDR_10_11,		0	65535	RO
165	HR_LAN_IPV6_GLOBAL_ADDR_12_13,		0	65535	RO
166	HR_LAN_IPV6_GLOBAL_ADDR_14_15,		0	65535	RO
167	HR_CHD1_LEVEL,	Logical level of digital input 1.	0	1	RO
168	HR_CHD2_LEVEL,	Logical level of digital input 2.	0	1	RO
169	HR_CHD3_LEVEL,	Logical level of digital input 3.	0	1	RO
170	HR_CHD4_LEVEL,	Logical level of digital input 4.	0	1	RO
171	HR_CHD5_LEVEL,	Logical level of digital input 5.	0	1	RO
172	HR_CHD6_LEVEL,	Logical level of digital input 6.	0	1	RO
173		Reserved.			
174	HR_CHD1_SETVALUE_H	Changes the value of the 32-bit counter for channel 1.	0	65535	RW
175	HR_CHD1_SETVALUE_L		0	65535	RW
176	HR_CHD2_SETVALUE_H	Changes the value of the 32-bit counter for channel 2.	0	65535	RW
177	HR_CHD2_SETVALUE_L		0	65535	RW
178	HR_CHD3_SETVALUE_H	Changes the value of the 32-bit counter for channel 3.	0	65535	RW
179	HR_CHD3_SETVALUE_L		0	65535	RW
180	HR_CHD4_SETVALUE_H	Changes the value of the 32-bit counter for channel 4.	0	65535	RW
181	HR_CHD4_SETVALUE_L,		0	65535	RW
182	HR_CHD5_SETVALUE_H	Changes the value of the 32-bit counter for channel 5.	0	65535	RW
183	HR_CHD5_SETVALUE_L		0	65535	RW
184	HR_CHD6_SETVALUE_H	Changes the value of the 32-bit counter for channel 6.	0	65535	RW
185	HR_CHD6_SETVALUE_L		0	65535	RW
186		Reserved.			
187	HR_SS_COLLECT_RECORD_MAX_QTTY	Maximum number of collections supported by memory.	1824	7096	RO
188	HR_SS_COLLECT_LAST_RECORD	Position of the last collection added to memory.	0	7096	RO
189	HR_SS_COLLECT_FIRST_ RECORD	Position of the first collection added to memory.	0	7096	RO
190	HR_SS_COLLECT_REQUESTED_ RECORD	Position of the collection requested for the reading.	0	7096	RW
191	HR_SS_COLLECT_TIMESTAMP_ UNIX_H	Timestamp of the requested collection in Unix format.	0	65535	RO
192	HR_SS_COLLECT_TIMESTAMP_ UNIX_L		0	65535	RO
193	HR_SS_COLLECT_TIMESTAMP_ MS	Timestamp of the requested collection in milliseconds.	0	65535	RO
194	HR_SS_COLLECT_CHD_EVENT_ INDEX	When an event occurs in the requested download, returns the index of the digital channel: 0 → No event. It is a periodic log. 1 → Event on channel 1. 2 → Event on channel 2. 3 → Event on channel 3. 4 → Event on channel 4. 5 → Event on channel 5. 6 → Event on channel 6.	0	6	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
195	HR_SS_COLLECT_CHD_EVENT_TYPE	When an event occurs in the requested download, returns the event type: 0 → No event. It is a periodic log. 1 → Falling edge event of the digital channel. 2 → Rising edge event of the digital channel.	0	2	RO
196	HR_SS_COLLECT_CHD1_VALUE_H	Value of digital channel 1 in the requested collection.	0	65535	RO
197	HR_SS_COLLECT_CHD1_VALUE_L		0	65535	RO
198	HR_SS_COLLECT_CHD2_VALUE_H	Value of digital channel 2 in the requested collection.	0	65535	RO
199	HR_SS_COLLECT_CHD2_VALUE_L		0	65535	RO
200	HR_SS_COLLECT_CHD3_VALUE_H	Value of digital channel 3 in the requested collection.	0	65535	RO
201	HR_SS_COLLECT_CHD3_VALUE_L		0	65535	RO
202	HR_SS_COLLECT_CHD4_VALUE_H	Value of digital channel 4 in the requested collection.	0	65535	RO
203	HR_SS_COLLECT_CHD4_VALUE_L		0	65535	RO
204	HR_SS_COLLECT_CHD5_VALUE_H	Value of digital channel 5 in the requested collection.	0	65535	RO
205	HR_SS_COLLECT_CHD5_VALUE_L		0	65535	RO
206	HR_SS_COLLECT_CHD6_VALUE_H	Value of digital channel 6 in the requested collection.	0	65535	RO
207	HR_SS_COLLECT_CHD6_VALUE_L		0	65535	RO
208	HR_SS_COLLECT_CH1_SENSE_USER_RANGE_H	Displays the sensor value in the user range of analog channel 1 (in Float).	0	65535	RO
209	HR_SS_COLLECT_CH1_SENSE_USER_RANGE_L		0	65535	RO
210	HR_SS_COLLECT_CH2_SENSE_USER_RANGE_H	Displays the sensor value in the user range of analog channel 2 (in Float).	0	65535	RO
211	HR_SS_COLLECT_CH2_SENSE_USER_RANGE_L		0	65535	RO
		Reserved.			
216	HR_SS_WIFI_RSSI_MIN	Minimum value indicated by the HR_WIFI_RSSI register. You can use the HR_RESET_DIAG_RSSI register to reset the value.	-120	-20	RO
217	HR_SS_WIFI_RSSI_MAX	Maximum value indicated by the HR_WIFI_RSSI register. You can use the HR_RESET_DIAG_RSSI register to reset the value.	-120	-20	RO
218	HR_SS_WIFI_RSSI_AVERAGE	Average value indicated by the HR_WIFI_RSSI register. You can use the HR_RESET_DIAG_RSSI register to reset the value.	-120	-20	RO
		Reserved.			
221	HR_RESET_COUNTER_WDT	Resets the system Watchdog diagnostic counters.	0	1	RW
222	HR_RESET_COUNTER_LOGS	Resets the system logs diagnostic counters.	0	1	RW
223	HR_RESET_DIAG_RSSI	Resets the minimum, maximum, and average signal quality (RSSI) measurement.	0	1	RW
		Reserved.			
227	HR_SS_CHR1_STATUS	Remote channel 1 status. 0 → Disabled channel. 1 → Action not allowed on the slave device. 2 → Register address not allowed. 3 → Query value not allowed. 4, 5, 6, 7, and 8 → Reserved. 9 → Communication timeout. 10 → CRC failure in the device response. 11 → Successful communication.	0	11	RO
228	HR_SS_CH_REMOTE_01_VALUE_	Reports the value of the last reading from remote	0x0000	0xFFFF	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
	HH	channel 1.			
229	HR_SS_CH_REMOTE_01_VALUE_HL	The layout in the registers follows the high first and depends on the type of data configured in the channel (whether it is 16 bits, 32 bits, or 64 bits).	0x0000	0xFFFF	RO
230	HR_SS_CH_REMOTE_01_VALUE_LH	If the channel is set to 16 bits, the value will be in the final _LL register.	0x0000	0xFFFF	RO
231	HR_SS_CH_REMOTE_01_VALUE_LL	If it is set to 32 bits, the data will be in the _LH and _LL registers. If it is set to 64 bits, the value will be in the 4 registers _HH, _HL, _LH, and _LL.	0x0000	0xFFFF	RO
232	HR_SS_CH_REMOTE_01_DOUBLE_HH	Remote channel 1 value in double 64-bit format. The layout of the double 64-bit data in the 4 Modbus registers follows high part first.	0x0000	0xFFFF	RO
233	HR_SS_CH_REMOTE_01_DOUBLE_HL		0x0000	0xFFFF	RO
234	HR_SS_CH_REMOTE_01_DOUBLE_LH		0x0000	0xFFFF	RO
235	HR_SS_CH_REMOTE_01_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
237	HR_SS_CHR2_STATUS	Remote channel 2 status.	0	11	RO
238	HR_SS_CH_REMOTE_02_VALUE_HH	Reports the value of the last reading from remote channel 2.	0x0000	0xFFFF	RO
239	HR_SS_CH_REMOTE_02_VALUE_HL		0x0000	0xFFFF	RO
240	HR_SS_CH_REMOTE_02_VALUE_LH		0x0000	0xFFFF	RO
241	HR_SS_CH_REMOTE_02_VALUE_LL		0x0000	0xFFFF	RO
242	HR_SS_CH_REMOTE_02_DOUBLE_HH	Remote channel 2 value in double 64-bit format.	0x0000	0xFFFF	RO
243	HR_SS_CH_REMOTE_02_DOUBLE_HL		0x0000	0xFFFF	RO
244	HR_SS_CH_REMOTE_02_DOUBLE_LH		0x0000	0xFFFF	RO
245	HR_SS_CH_REMOTE_02_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
247	HR_SS_CHR3_STATUS	Remote channel 3 status.	0	11	RO
248	HR_SS_CH_REMOTE_03_VALUE_HH	Reports the value of the last reading from remote channel 3.	0x0000	0xFFFF	RO
249	HR_SS_CH_REMOTE_03_VALUE_HL		0x0000	0xFFFF	RO
250	HR_SS_CH_REMOTE_03_VALUE_LH		0x0000	0xFFFF	RO
251	HR_SS_CH_REMOTE_03_VALUE_LL		0x0000	0xFFFF	RO
252	HR_SS_CH_REMOTE_03_DOUBLE_HH	Remote channel 3 value in double 64-bit format.	0x0000	0xFFFF	RO
253	HR_SS_CH_REMOTE_03_DOUBLE_HL		0x0000	0xFFFF	RO
254	HR_SS_CH_REMOTE_03_DOUBLE_LH		0x0000	0xFFFF	RO
255	HR_SS_CH_REMOTE_03_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
247	HR_SS_CHR4_STATUS	Remote channel 4 status.	0	11	RO
258	HR_SS_CH_REMOTE_04_VALUE_HH	Reports the value of the last reading from remote channel 4.	0x0000	0xFFFF	RO
259	HR_SS_CH_REMOTE_04_VALUE_HL		0x0000	0xFFFF	RO
260	HR_SS_CH_REMOTE_04_VALUE_LH		0x0000	0xFFFF	RO
261	HR_SS_CH_REMOTE_04_VALUE_LL		0x0000	0xFFFF	RO
262	HR_SS_CH_REMOTE_04_DOUBLE_HH	Remote channel 4 value in double 64-bit format.	0x0000	0xFFFF	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
263	HR_SS_CH_REMOTE_04_DOUBLE_HL		0x0000	0xFFFF	RO
264	HR_SS_CH_REMOTE_04_DOUBLE_LH		0x0000	0xFFFF	RO
265	HR_SS_CH_REMOTE_04_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
267	HR_SS_CHR5_STATUS	Remote channel 5 status.	0	11	RO
268	HR_SS_CH_REMOTE_05_VALUE_HH	Reports the value of the last reading from remote channel 5.	0x0000	0xFFFF	RO
269	HR_SS_CH_REMOTE_05_VALUE_HL		0x0000	0xFFFF	RO
270	HR_SS_CH_REMOTE_05_VALUE_LH		0x0000	0xFFFF	RO
271	HR_SS_CH_REMOTE_05_VALUE_LL		0x0000	0xFFFF	RO
272	HR_SS_CH_REMOTE_05_DOUBLE_HH	Remote channel 5 value in double 64-bit format.	0x0000	0xFFFF	RO
273	HR_SS_CH_REMOTE_05_DOUBLE_HL		0x0000	0xFFFF	RO
274	HR_SS_CH_REMOTE_05_DOUBLE_LH		0x0000	0xFFFF	RO
275	HR_SS_CH_REMOTE_05_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
277	HR_SS_CHR6_STATUS	Remote channel 6 status.	0	11	RO
278	HR_SS_CH_REMOTE_06_VALUE_HH	Reports the value of the last reading from remote channel 6.	0x0000	0xFFFF	RO
279	HR_SS_CH_REMOTE_06_VALUE_HL		0x0000	0xFFFF	RO
280	HR_SS_CH_REMOTE_06_VALUE_LH		0x0000	0xFFFF	RO
281	HR_SS_CH_REMOTE_06_VALUE_LL		0x0000	0xFFFF	RO
282	HR_SS_CH_REMOTE_06_DOUBLE_HH	Remote channel 6 value in double 64-bit format.	0x0000	0xFFFF	RO
283	HR_SS_CH_REMOTE_06_DOUBLE_HL		0x0000	0xFFFF	RO
284	HR_SS_CH_REMOTE_06_DOUBLE_LH		0x0000	0xFFFF	RO
285	HR_SS_CH_REMOTE_06_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
287	HR_SS_CHR7_STATUS	Remote channel 7 status.	0	11	RO
288	HR_SS_CH_REMOTE_07_VALUE_HH	Reports the value of the last reading from remote channel 7.	0x0000	0xFFFF	RO
289	HR_SS_CH_REMOTE_07_VALUE_HL		0x0000	0xFFFF	RO
290	HR_SS_CH_REMOTE_07_VALUE_LH		0x0000	0xFFFF	RO
291	HR_SS_CH_REMOTE_07_VALUE_LL		0x0000	0xFFFF	RO
292	HR_SS_CH_REMOTE_07_DOUBLE_HH	Remote channel 7 value in double 64-bit format.	0x0000	0xFFFF	RO
293	HR_SS_CH_REMOTE_07_DOUBLE_HL		0x0000	0xFFFF	RO
294	HR_SS_CH_REMOTE_07_DOUBLE_LH		0x0000	0xFFFF	RO
295	HR_SS_CH_REMOTE_07_DOUBLE_LL		0x0000	0xFFFF	RO
		Reserved.			
297	HR_SS_CHR8_STATUS	Remote channel 8 status.	0	11	RO
298	HR_SS_CH_REMOTE_08_VALUE_HH	Reports the value of the last reading from remote channel 8.	0x0000	0xFFFF	RO

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
299	HR_SS_CH_REMOTE_08_VALUE_HL		0x0000	0xFFFF	RO
300	HR_SS_CH_REMOTE_08_VALUE_LH		0x0000	0xFFFF	RO
301	HR_SS_CH_REMOTE_08_VALUE_LL		0x0000	0xFFFF	RO
302	HR_SS_CH_REMOTE_08_DOUBLE_HH	Remote channel 8 value in double 64-bit format.	0x0000	0xFFFF	RO
303	HR_SS_CH_REMOTE_08_DOUBLE_HL		0x0000	0xFFFF	RO
304	HR_SS_CH_REMOTE_08_DOUBLE_LH		0x0000	0xFFFF	RO
305	HR_SS_CH_REMOTE_08_DOUBLE_LL		0x0000	0xFFFF	RO

Table 16

12.2.1 TABLE OF REGISTERS: CONFIGURATION

DigiRail IoT supports the following configuration registers:

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1000	HR_CS_SETTING_RESTORE_DEFAULT	Allows you to return all configuration parameters to the factory settings.	0	1	RW
1001	HR_CS_SETTING_PUBLISH_INTERVAL_S	Allows you to configure the publishing interval of the device.	1	65535	RW
1002	HR_CS_APPLY_CONFIGURATION	Allows you to instantly apply the settings.	0	1	RW
1003	HR_CS_LOCATION_0	Allows you to define the location of the device. Each register corresponds to two characters.	0x0000	0xFFFF	RW
1004	HR_CS_LOCATION_1		0x0000	0xFFFF	RW
1005	HR_CS_LOCATION_2		0x0000	0xFFFF	RW
1006	HR_CS_LOCATION_3		0x0000	0x00FF	RW
1007	HR_CS_LOCATION_4		0x0000	0xFFFF	RW
1008	HR_CS_LOCATION_5		0x0000	0xFFFF	RW
1009	HR_CS_LOCATION_6		0x0000	0xFFFF	RW
1010	HR_CS_LOCATION_7		0x0000	0xFFFF	RW
1011	HR_CS_LOCATION_8		0x0000	0xFFFF	RW
1012	HR_CS_LOCATION_9		0x0000	0xFFFF	RW
1013	HR_CS_LOCATION_10		0x0000	0xFFFF	RW
1014	HR_CS_LOCATION_11		0x0000	0xFFFF	RW
1015	HR_CS_LOCATION_12		0x0000	0xFFFF	RW
1016	HR_CS_LOCATION_13		0x0000	0xFFFF	RW
1017	HR_CS_LOCATION_14		0x0000	0xFFFF	RW
1018	HR_CS_LOCATION_15		0x0000	0xFFFF	RW
1019	HR_CS_LOCATION_16		0x0000	0xFFFF	RW
1020	HR_CS_LOCATION_17		0x0000	0xFFFF	RW
1021	HR_CS_LOCATION_18		0x0000	0xFFFF	RW
1022	HR_CS_LOCATION_19		0x0000	0xFFFF	RW
1023	HR_CS_SETTING_ENABLE_ALTERNATIVE_PUB_INTERVAL	Allows you to enable an alternative publishing interval whenever there are connection problems.	0	1	RW
1024	HR_CS_SETTING_ALTERNATIVE_PUB_INTERVAL_S	Allows you to configure an alternative publishing interval whenever there are connection problems.	60	65535	RW
1025	HR_CS_SETTING_FORCE_RESET_COUNTERS	Allows you to reset the digital channel counters, regardless of the channel's permission.	0	1	RW
		Reserved.			

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1027	HR_CS_SETTING_TITLE_1	Allows you to configure a name for the device. Each register corresponds to two characters.	0x0000	0xFFFF	RW
1028	HR_CS_SETTING_TITLE_2		0x0000	0xFFFF	RW
1029	HR_CS_SETTING_TITLE_3		0x0000	0xFFFF	RW
1030	HR_CS_SETTING_TITLE_4		0x0000	0xFFFF	RW
1031	HR_CS_SETTING_TITLE_5		0x0000	0xFFFF	RW
1032	HR_CS_SETTING_TITLE_6		0x0000	0xFFFF	RW
1033	HR_CS_SETTING_TITLE_7		0x0000	0xFFFF	RW
1034	HR_CS_SETTING_TITLE_8		0x0000	0xFFFF	RW
1035	HR_CS_SETTING_TITLE_9		0x0000	0xFFFF	RW
1036	HR_CS_SETTING_TITLE_10		0x0000	0xFFFF	RW
		Reserved.			
1038	HR_CS_SETTING_GMT	Allows you to configure GMT.	0x0000	0xFFFF	RW
1039	HR_CS_SETTING_YEAR	Allows you to configure the year (RTC: GMT 0).	2016	2080	RW
1040	HR_CS_SETTING_MONTH	Allows you to configure the month (RTC: GMT 0).	1	12	RW
1041	HR_CS_SETTING_DAY	Allows you to configure the day (RTC: GMT 0).	1	31	RW
1042	HR_CS_SETTING_HOUR	Allows you to configure the hour (RTC: GMT 0).	0	23	RW
1043	HR_CS_SETTING_MINUTE	Allows you to configure the minute (RTC: GMT 0).	0	59	RW
1044	HR_CS_SETTING_SECOND	Allows you to configure the second (RTC: GMT 0).	0	59	RW
		Reserved.			
1046	HR_CS_CHD_ADD_COUNTER_AT_EVENT	Allows you to add the digital channel count value to the event log.	0	1	RW
1047	HR_CS_CHD_RESET_TYPE	Allows you to define the reset mode of the digital counters: 0 → Daily. 1 → Weekly. 2 → Monthly	0	2	RW
1048	HR_CS_CHD_RESET_DAY	Allows you to configure the reset day for the digital counters according to what was configured in register 1060.	1	31	RW
1049	HR_CS_CHD_RESET_HOUR	Allows you to configure the reset time for the digital counters according to what was configured in register 1060.	0	23	RW
1050	HR_CS_CHD_RESET_MINUTE	Allows you to configure the reset minute for the digital counters according to what was configured in register 1060.	0	59	RW
1051	HR_CS_CHD_RESET_SECOND	Allows you to configure the reset second for the digital counters according to what was configured in register 1060.	0	59	RW
1052	HR_CS_CHD_RESET_WEEK_DAY	Allows you to configure the reset week for the digital counters according to what was configured in register 1060.	1	7	RW
1053	HR_CS_CHD1_ENABLED	Allows you to enable digital channel 1.	0	1	RW
1054	HR_CS_CHD1_COUNTING_MODE	Allows you to configure the counting mode for digital channel 1: 0 → Not configured. 1 → Counter. 2 → Event.	0	2	RW
1055	HR_CS_CHD1_SENSOR_TYPE	Allows you to configure the sensor type of digital channel 1: 0 → Not configured. 1 → PNP. 2 → NPN. 3 → Dry contact.	0	3	RW
1056	HR_CS_CHD1_COUNTING_EDGE	Allows you to configure the counting edge of digital channel 1: 1 → Rising edge. 2 → Falling edge. 3 → Both edges.	1	3	RW
1057	HR_CS_CHD1_DEBOUNCE_TIME_ms	Allows you to configure the Debounce time of the digital channel 1 for the Dry Contact sensor (in milliseconds).	0	60000	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
		Reserved.			
1060	HR_CS_CHD1_RESET_MODE	Allows you to configure the reset mode of the accumulators of digital channel 1: Bit 0 → Overflow. Bit 1 → Calendar. Bit 2 → Protocol.	0	2	RW
1061	HR_CS_CHD1_DEBOUNCE_ ENABLE	Allows you to enable Debounce for digital channel 1.	0	1	RW
1062	HR_CS_CHD1_CHANGE_VALUE_ ENABLE	Allows you to enable the permission to change the digital channel 1 count value via Modbus-TCP or MQTT protocols.	0	1	RW
		Reserved.			
1067	HR_CS_CHD2_ENABLED	Allows you to enable digital channel 2.	0	1	RW
1068	HR_CS_CHD2_COUNTING_MODE	Allows you to configure the counting mode for digital channel 2: 0 → Not configured. 1 → Counter. 2 → Event.	0	2	RW
1069	HR_CS_CHD2_SENSOR_TYPE	Allows you to configure the sensor type of digital channel 2: 0 → Not configured. 1 → PNP. 2 → NPN. 3 → Dry contact.	0	3	RW
1070	HR_CS_CHD2_COUNTING_EDGE	Allows you to configure the counting edge of digital channel 2: 1 → Rising edge. 2 → Falling edge. 3 → Both edges.	1	3	RW
1071	HR_CS_CHD2_DEBOUNCE_TIME_ ms	Allows you to configure the Debounce time of the digital channel 2 for the Dry Contact sensor (in milliseconds).	0	60000	RW
		Reserved.			
1074	HR_CS_CHD2_RESET_MODE	Allows you to configure the reset mode of the accumulators of digital channel 2: Bit 0 → Overflow. Bit 1 → Calendar. Bit 2 → Protocol.	0	3	RW
1075	HR_CS_CHD2_DEBOUNCE_ ENABLE	Allows you to enable Debounce for digital channel 2.	0	1	RW
1076	HR_CS_CHD2_CHANGE_VALUE_ ENABLE	Allows you to enable the permission to change the digital channel 2 count value via Modbus-TCP or MQTT protocols.	0	1	RW
		Reserved.			
1081	HR_CS_CHD3_ENABLED	Allows you to enable digital channel 3.	0	1	RW
1082	HR_CS_CHD3_COUNTING_MODE	Allows you to configure the counting mode for digital channel 3: 0 → Not configured. 1 → Counter. 2 → Event.	0	2	RW
1083	HR_CS_CHD3_SENSOR_TYPE	Allows you to configure the sensor type of digital channel 3: 0 → Not configured. 1 → PNP. 2 → NPN. 3 → Dry contact.	0	3	RW
1084	HR_CS_CHD3_COUNTING_EDGE	Allows you to configure the counting edge of digital channel 3: 1 → Rising edge. 2 → Falling edge. 3 → Both edges.	1	3	RW
1085	HR_CS_CHD3_DEBOUNCE_TIME_ ms	Allows you to configure the Debounce time of the digital channel 3 for the Dry Contact sensor (in milliseconds).	0	60000	RW
		Reserved.			

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1088	HR_CS_CHD3_RESET_MODE	Allows you to configure the reset mode of the accumulators of digital channel 3: Bit 0 → Overflow. Bit 1 → Calendar. Bit 2 → Protocol.	0	2	RW
1089	HR_CS_CHD3_DEBOUNCE_ ENABLE	Allows you to enable Debounce for digital channel 3.	0	1	RW
1090	HR_CS_CHD3_CHANGE_VALUE_ ENABLE	Allows you to enable the permission to change the digital channel 3 count value via Modbus-TCP or MQTT protocols.	0	1	RW
		Reserved.			
1095	HR_CS_CHD4_ENABLED	Allows you to enable digital channel 4.	0	1	RW
1096	HR_CS_CHD4_COUNTING_MODE	Allows you to configure the counting mode for digital channel 4: 0 → Not configured. 1 → Counter. 2 → Event.	0	2	RW
1097	HR_CS_CHD4_SENSOR_TYPE	Allows you to configure the sensor type of digital channel 4: 0 → Not configured. 1 → PNP. 2 → NPN. 3 → Dry contact.	0	3	RW
1098	HR_CS_CHD4_COUNTING_EDGE	Allows you to configure the counting edge of digital channel 4: 1 → Rising edge. 2 → Falling edge. 3 → Both edges.	1	3	RW
1099	HR_CS_CHD4_DEBOUNCE_TIME_ ms	Allows you to configure the Debounce time of the digital channel 4 for the Dry Contact sensor (in milliseconds).	0	60000	RW
		Reserved.			
1102	HR_CS_CHD4_RESET_MODE	Allows you to configure the reset mode of the accumulators of digital channel 4: Bit 0 → Overflow. Bit 1 → Calendar. Bit 2 → Protocol.	0	2	RW
1103	HR_CS_CHD4_DEBOUNCE_ ENABLE	Allows you to enable Debounce for digital channel 4.	0	1	RW
1104	HR_CS_CHD4_CHANGE_VALUE_ ENABLE	Allows you to enable the permission to change the digital channel 4 count value via Modbus-TCP or MQTT protocols.	0	1	RW
		Reserved.			
1109	HR_CS_CHD5_ENABLED	Allows you to enable digital channel 5.	0	1	RW
1110	HR_CS_CHD5_COUNTING_MODE	Allows you to configure the counting mode for digital channel 5: 0 → Not configured. 1 → Counter. 2 → Event.	0	2	RW
1111	HR_CS_CHD5_SENSOR_TYPE	Allows you to configure the sensor type of digital channel 5: 0 → Not configured. 1 → PNP. 2 → NPN. 3 → Dry contact.	0	3	RW
1112	HR_CS_CHD5_COUNTING_EDGE	Allows you to configure the counting edge of digital channel 5: 1 → Rising edge. 2 → Falling edge. 3 → Both edges.	1	3	RW
1113	HR_CS_CHD5_DEBOUNCE_TIME_ ms	Allows you to configure the Debounce time of the digital channel 5 for the Dry Contact sensor (in milliseconds).	0	60000	RW
		Reserved.			
1116	HR_CS_CHD5_RESET_MODE	Allows you to configure the reset mode of the	0	2	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
		accumulators of digital channel 5: Bit 0 → Overflow. Bit 1 → Calendar. Bit 2 → Protocol.			
1117	HR_CS_CHD5_DEBOUNCE_ENABLE	Allows you to enable Debounce for digital channel 5.	0	1	RW
1118	HR_CS_CHD5_CHANGE_VALUE_ENABLE	Allows you to enable the permission to change the digital channel 4 count value via Modbus-TCP or MQTT protocols.	0	1	RW
		Reserved.			
1123	HR_CS_CHD6_ENABLED	Allows you to enable digital channel 6.	0	1	RW
1124	HR_CS_CHD6_COUNTING_MODE	Allows you to configure the counting mode for digital channel 6: 0 → Not configured. 1 → Counter. 2 → Event.	0	2	RW
1125	HR_CS_CHD6_SENSOR_TYPE	Allows you to configure the sensor type of digital channel 6: 0 → Not configured. 1 → PNP. 2 → NPN. 3 → Dry contact.	0	3	RW
1126	HR_CS_CHD6_COUNTING_EDGE	Allows you to configure the counting edge of digital channel 6: 1 → Rising edge. 2 → Falling edge. 3 → Both edges.	1	3	RW
1127	HR_CS_CHD6_DEBOUNCE_TIME_ms	Allows you to configure the Debounce time of the digital channel 6 for the Dry Contact sensor (in milliseconds).	0	60000	RW
		Reserved.			
1130	HR_CS_CHD6_RESET_MODE	Allows you to configure the reset mode of the accumulators of digital channel 6: Bit 0 → Overflow. Bit 1 → Calendar. Bit 2 → Protocol.	0	2	RW
1131	HR_CS_CHD6_DEBOUNCE_ENABLE	Allows you to enable Debounce for digital channel 6.	0	1	RW
1132	HR_CS_CHD6_CHANGE_VALUE_ENABLE	Allows you to enable the permission to change the digital channel 6 count value via Modbus-TCP or MQTT protocols.	0	1	RW
		Reserved.			
1138	HR_CS_CH1_ENABLE	Allows you to enable analog channel 1.	0	1	RW
		Reserved.			
1140	HR_CS_CH1_SENSOR_TYPE	Allows you to configure the sensor type of digital channel 1: 0 → Not configured. 0 → 0-5 V. 2 → 0-10 V. 3 → 0-20 mA. 4 → 4-20 mA.	0	4	RW
		Reserved.			
1142	HR_CS_CH1_RANGE_MIN_HIGH	Allows you to configure the minimum limit of analog channel 1.	0	0xFFFF	RW
1143	HR_CS_CH1_RANGE_MIN_LOW		0	0xFFFF	RW
1144	HR_CS_CH1_RANGE_MAX_HIGH	Allows you to configure the maximum limit of analog channel 1.	0	0xFFFF	RW
1145	HR_CS_CH1_RANGE_MAX_LOW		0	0xFFFF	RW
1146	HR_CS_CH1_DECIMAL_POINT	Allows you to configure the decimal place of analog channel 1 (fixed point for display and memory register). 0 → No decimal places. 1 → One decimal place. 2 → Two decimal places.	0	2	RW
1147	HR_CS_CH1_ERROR_VALUE_	Allows you to configure a value for error indication	0	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
	HIGH	on analog channel 1.			
1148	HR_CS_CH1_ERROR_VALUE_ LOW		0	0xFFFF	RW
		Reserved.			
1158	HR_CS_CH2_ENABLE	Allows you to enable analog channel 2.	0	1	RW
		Reserved.			
1160	HR_CS_CH2_SENSOR_TYPE	Allows you to configure the sensor type of digital channel 2: 0 → Not configured. 0 → 0-5 V. 2 → 0-10 V. 3 → 0-20 mA. 4 → 4-20 mA.	0	4	RW
		Reserved.			
1162	HR_CS_CH2_RANGE_MIN_HIGH	Allows you to configure the minimum limit of analog channel 2.	0	0xFFFF	RW
1163	HR_CS_CH2_RANGE_MIN_LOW		0	0xFFFF	RW
1164	HR_CS_CH2_RANGE_MAX_HIGH	Allows you to configure the maximum limit of analog channel 2.	0	0xFFFF	RW
1165	HR_CS_CH2_RANGE_MAX_LOW		0	0xFFFF	RW
1166	HR_CS_CH2_DECIMAL_POINT	Allows you to configure the decimal place of analog channel 1 (fixed point for display and memory register). 0 → No decimal places. 1 → One decimal place. 2 → Two decimal places.	0	2	RW
1167	HR_CS_CH2_ERROR_VALUE_ HIGH	Allows you to configure a value for error indication on analog channel 2.	0	0xFFFF	RW
1168	HR_CS_CH2_ERROR_VALUE_ LOW		0	0xFFFF	RW
		Reserved.			
1172	HR_CS_MODBUS_TCP_ENABLE	Allows you to enable the Modbus-TCP protocol.	0	1	RW
1173	HR_CS_MODBUS_TCP_PORT	Allows you to configure the Modbus-TCP protocol communication port.	0	0xFFFF	RW
1174	HR_CS_MODBUS_TCP_ADDRESS	Allows you to configure the Modbus address for the device. For different values, it will forward the data to RS485, acting as a Gateway.	1	255	RW
1175	HR_CS_RS485_BAUD_RATE,	Allows you to configure the RS485 interface Baud Rate: 0 → 1200. 1 → 2400. 2 → 4800. 3 → 9600. 4 → 19200. 5 → 38400. 6 → 57600. 7 → 115200.	0	7	RW
1176	HR_CS_RS485_STOP_BITS,	Allows you to configure the RS485 interface Stop Bits: 0 → 1 Stop Bit. 1 → 2 Stop Bits.	0	1	RW
1177	HR_CS_RS485_PARITY,	Allows you to configure the RS485 interface parity: 0 → No parity. 1 → Odd parity. 2 → Even parity.	0	2	RW
1178	HR_CS_RS485_TIMEOUT	Allows you to configure a timeout value for the connection (in milliseconds).	0	65535	RW
		Reserved.			
1180	HR_CS_ETH_IP_GET_ADRESS	Allows you to define how the device will obtain an IP address: 0 → Static. 1 → DHCP.	0	1	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
		Reserved.			
1182	HR_CS_ETH_IPV4_ADDR_0_1	Allows you to configure the device IPv4 address. Two octets per register.	0	65535	RW
1183	HR_CS_ETH_IPV4_ADDR_2_3	Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RW
1184	HR_CS_ETH_IPV4_MASK_ADDR_0_1	Allows you to define the network mask. Two octets per register.	0	65535	RW
1185	HR_CS_ETH_IPV4_MASK_ADDR_2_3	Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RW
1186	HR_CS_ETH_IPV4_GATEWAY_ADDR_0_1	Allows you to configure the network Gateway address. Two octets per register.	0	65535	RW
1187	HR_CS_ETH_IPV4_GATEWAY_ADDR_2_3	Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RW
1188	HR_CS_ETH_IPV4_DNS_ADDR_0_1	Allows you to configure the DNS server IP. Two octets per register.	0	65535	RW
1189	HR_CS_ETH_IPV4_DNS_ADDR_2_3	Dec 0 . Dec 1 . Dec 2 . Dec 3	0	65535	RW
1190	HR_CS_ETH_IPV6_PREFIX	Allows you to configure the IPv6 prefix.	0	128	RW
1191	HR_CS_ETH_IPV6_ADDR_0_1,	Allows you to configure the IPv6 – Local address. Hexadecimal format. 0_1 : 2_3 : 4_5 : 6_7 : 8_9 : 10_11 : 12_13 : 14_15	0	65535	RW
1192	HR_CS_ETH_IPV6_ADDR_2_3,		0	65535	RW
1193	HR_CS_ETH_IPV6_ADDR_4_5,		0	65535	RW
1194	HR_CS_ETH_IPV6_ADDR_6_7,		0	65535	RW
1195	HR_CS_ETH_IPV6_ADDR_8_9,		0	65535	RW
1196	HR_CS_ETH_IPV6_ADDR_10_11,		0	65535	RW
1197	HR_CS_ETH_IPV6_ADDR_12_13,		0	65535	RW
1198	HR_CS_ETH_IPV6_ADDR_14_15,		0	65535	RW
1199	HR_CS_ETH_IPV6_DNS_ADDR_0_1,	Allows you to configure the IPv6 – Global address. Hexadecimal format. 0_1 : 2_3 : 4_5 : 6_7 : 8_9 : 10_11 : 12_13 : 14_15	0	65535	RW
1200	HR_CS_ETH_IPV6_DNS_ADDR_2_3,		0	65535	RW
1201	HR_CS_ETH_IPV6_DNS_ADDR_4_5,		0	65535	RW
1202	HR_CS_ETH_IPV6_DNS_ADDR_6_7,		0	65535	RW
1203	HR_CS_ETH_IPV6_DNS_ADDR_8_9,		0	65535	RW
1204	HR_CS_ETH_IPV6_DNS_ADDR_10_11,		0	65535	RW
1205	HR_CS_ETH_IPV6_DNS_ADDR_12_13,		0	65535	RW
1206	HR_CS_ETH_IPV6_DNS_ADDR_14_15,		0	65535	RW
1207	HR_CS_WIFI_ROUTER_TITLE_1	Allows you to define the Wi-Fi network SSID. Each register corresponds to two characters.	0x0000	0xFFFF	RW
1208	HR_CS_WIFI_ROUTER_TITLE_2		0x0000	0xFFFF	RW
1209	HR_CS_WIFI_ROUTER_TITLE_3		0x0000	0xFFFF	RW
1210	HR_CS_WIFI_ROUTER_TITLE_4		0x0000	0xFFFF	RW
1211	HR_CS_WIFI_ROUTER_TITLE_5		0x0000	0xFFFF	RW
1212	HR_CS_WIFI_ROUTER_TITLE_6		0x0000	0xFFFF	RW
1213	HR_CS_WIFI_ROUTER_TITLE_7		0x0000	0xFFFF	RW
1214	HR_CS_WIFI_ROUTER_TITLE_8		0x0000	0xFFFF	RW
1215	HR_CS_WIFI_ROUTER_TITLE_9		0x0000	0xFFFF	RW
1216	HR_CS_WIFI_ROUTER_TITLE_10		0x0000	0xFFFF	RW
1217	HR_CS_WIFI_ROUTER_TITLE_11		0x0000	0xFFFF	RW
1218	HR_CS_WIFI_ROUTER_TITLE_12		0x0000	0xFFFF	RW
1219	HR_CS_WIFI_ROUTER_TITLE_13		0x0000	0xFFFF	RW
1220	HR_CS_WIFI_ROUTER_TITLE_14		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1221	HR_CS_WIFI_ROUTER_TITLE_15		0x0000	0xFFFF	RW
1222	HR_CS_WIFI_ROUTER_TITLE_16		0x0000	0xFFFF	RW
		Reserved.			
1224	HR_CS_WIFI_ROUTER_ PASSWORD_1	Allows you to define the Wi-Fi network password. Each register corresponds to two characters.	0x0000	0xFFFF	RW
1225	HR_CS_WIFI_ROUTER_ PASSWORD_2		0x0000	0xFFFF	RW
1226	HR_CS_WIFI_ROUTER_ PASSWORD_3		0x0000	0xFFFF	RW
1227	HR_CS_WIFI_ROUTER_ PASSWORD_4		0x0000	0xFFFF	RW
1228	HR_CS_WIFI_ROUTER_ PASSWORD_5		0x0000	0xFFFF	RW
1229	HR_CS_WIFI_ROUTER_ PASSWORD_6		0x0000	0xFFFF	RW
1230	HR_CS_WIFI_ROUTER_ PASSWORD_7		0x0000	0xFFFF	RW
1231	HR_CS_WIFI_ROUTER_ PASSWORD_8		0x0000	0xFFFF	RW
1232	HR_CS_WIFI_ROUTER_ PASSWORD_9		0x0000	0xFFFF	RW
1233	HR_CS_WIFI_ROUTER_ PASSWORD_10		0x0000	0xFFFF	RW
1234	HR_CS_WIFI_ROUTER_ PASSWORD_11		0x0000	0xFFFF	RW
1235	HR_CS_WIFI_ROUTER_ PASSWORD_12		0x0000	0xFFFF	RW
1236	HR_CS_WIFI_ROUTER_ PASSWORD_13		0x0000	0xFFFF	RW
1237	HR_CS_WIFI_ROUTER_ PASSWORD_14		0x0000	0xFFFF	RW
1238	HR_CS_WIFI_ROUTER_ PASSWORD_15		0x0000	0xFFFF	RW
1239	HR_CS_WIFI_ROUTER_ PASSWORD_16		0x0000	0xFFFF	RW
1240	HR_CS_WIFI_ROUTER_ PASSWORD_17		0x0000	0xFFFF	RW
1241	HR_CS_WIFI_ROUTER_ PASSWORD_18		0x0000	0xFFFF	RW
1242	HR_CS_WIFI_ROUTER_ PASSWORD_19		0x0000	0xFFFF	RW
1243	HR_CS_WIFI_ROUTER_ PASSWORD_20		0x0000	0xFFFF	RW
1244	HR_CS_WIFI_ROUTER_ PASSWORD_21		0x0000	0xFFFF	RW
		Reserved.			
1246	HR_CS_MQTT_ENABLE	Allows you to enable the MQTT protocol.	0	1	RW
1247	HR_CS_MQTT_QOS	Allows you to configure the QoS for sending messages.	0	1	RW
1248	HR_CS_MQTT_PORT	Allows you to configure the port to be used for the MQTT protocol.	0	65535	RW
1249	HR_CS_MQTT_ROUTER_TITLE_1	Allows you to enter the Broker user. Each register corresponds to two characters.	0x0000	0xFFFF	RW
1250	HR_CS_MQTT_ROUTER_TITLE_2		0x0000	0xFFFF	RW
1251	HR_CS_MQTT_ROUTER_TITLE_3		0x0000	0xFFFF	RW
1252	HR_CS_MQTT_ROUTER_TITLE_4		0x0000	0xFFFF	RW
1253	HR_CS_MQTT_ROUTER_TITLE_5		0x0000	0xFFFF	RW
1254	HR_CS_MQTT_ROUTER_TITLE_6		0x0000	0xFFFF	RW
1255	HR_CS_MQTT_ROUTER_TITLE_7		0x0000	0xFFFF	RW
1256	HR_CS_MQTT_ROUTER_TITLE_8		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1257	HR_CS_MQTT_ROUTER_TITLE_9		0x0000	0xFFFF	RW
1258	HR_CS_MQTT_ROUTER_TITLE_10		0x0000	0xFFFF	RW
1259	HR_CS_MQTT_ROUTER_TITLE_11		0x0000	0xFFFF	RW
1260	HR_CS_MQTT_ROUTER_TITLE_12		0x0000	0xFFFF	RW
1261	HR_CS_MQTT_ROUTER_TITLE_13		0x0000	0xFFFF	RW
1262	HR_CS_MQTT_ROUTER_TITLE_14		0x0000	0xFFFF	RW
1263	HR_CS_MQTT_ROUTER_TITLE_15		0x0000	0xFFFF	RW
1264	HR_CS_MQTT_ROUTER_TITLE_16		0x0000	0xFFFF	RW
		Reserved.			
1266	HR_CS_MQTT_BROKER_ PASSWORD_1		0x0000	0xFFFF	RW
1267	HR_CS_MQTT_BROKER_ PASSWORD_2		0x0000	0xFFFF	RW
1268	HR_CS_MQTT_BROKER_ PASSWORD_3		0x0000	0xFFFF	RW
1269	HR_CS_MQTT_BROKER_ PASSWORD_4		0x0000	0xFFFF	RW
1270	HR_CS_MQTT_BROKER_ PASSWORD_5		0x0000	0xFFFF	RW
1271	HR_CS_MQTT_BROKER_ PASSWORD_6		0x0000	0xFFFF	RW
1272	HR_CS_MQTT_BROKER_ PASSWORD_7		0x0000	0xFFFF	RW
1273	HR_CS_MQTT_BROKER_ PASSWORD_8		0x0000	0xFFFF	RW
1274	HR_CS_MQTT_BROKER_ PASSWORD_9		0x0000	0xFFFF	RW
1275	HR_CS_MQTT_BROKER_ PASSWORD_10		0x0000	0xFFFF	RW
1276	HR_CS_MQTT_BROKER_ PASSWORD_11	Allows you to configure the Broker password. Each register corresponds to two characters.	0x0000	0xFFFF	RW
1277	HR_CS_MQTT_BROKER_ PASSWORD_12		0x0000	0xFFFF	RW
1278	HR_CS_MQTT_BROKER_ PASSWORD_13		0x0000	0xFFFF	RW
1279	HR_CS_MQTT_BROKER_ PASSWORD_14		0x0000	0xFFFF	RW
1280	HR_CS_MQTT_BROKER_ PASSWORD_15		0x0000	0xFFFF	RW
1281	HR_CS_MQTT_BROKER_ PASSWORD_16		0x0000	0xFFFF	RW
1282	HR_CS_MQTT_BROKER_ PASSWORD_17		0x0000	0xFFFF	RW
1283	HR_CS_MQTT_BROKER_ PASSWORD_18		0x0000	0xFFFF	RW
1284	HR_CS_MQTT_BROKER_ PASSWORD_19		0x0000	0xFFFF	RW
1285	HR_CS_MQTT_BROKER_ PASSWORD_20		0x0000	0xFFFF	RW
1286	HR_CS_MQTT_BROKER_ PASSWORD_21		0x0000	0xFFFF	RW
		Reserved.			
1288	HR_CS_MQTT_BROKER_MQTT_IP _URL_1		0x0000	0xFFFF	RW
1289	HR_CS_MQTT_BROKER_MQTT_IP _URL_2	Allows you to define the IP or URL of the Broker. Each register corresponds to two characters.	0x0000	0xFFFF	RW
1290	HR_CS_MQTT_BROKER_MQTT_IP _URL_3		0x0000	0xFFFF	RW
1291	HR_CS_MQTT_BROKER_MQTT_IP _URL_4		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1292	HR_CS_MQTT_BROKER_MQTT_IP_URL_5		0x0000	0xFFFF	RW
1293	HR_CS_MQTT_BROKER_MQTT_IP_URL_6		0x0000	0xFFFF	RW
1294	HR_CS_MQTT_BROKER_MQTT_IP_URL_7		0x0000	0xFFFF	RW
1295	HR_CS_MQTT_BROKER_MQTT_IP_URL_8		0x0000	0xFFFF	RW
1296	HR_CS_MQTT_BROKER_MQTT_IP_URL_9		0x0000	0xFFFF	RW
1297	HR_CS_MQTT_BROKER_MQTT_IP_URL_10		0x0000	0xFFFF	RW
1298	HR_CS_MQTT_BROKER_MQTT_IP_URL_11		0x0000	0xFFFF	RW
1299	HR_CS_MQTT_BROKER_MQTT_IP_URL_12		0x0000	0xFFFF	RW
1300	HR_CS_MQTT_BROKER_MQTT_IP_URL_13		0x0000	0xFFFF	RW
1301	HR_CS_MQTT_BROKER_MQTT_IP_URL_14		0x0000	0xFFFF	RW
1302	HR_CS_MQTT_BROKER_MQTT_IP_URL_15		0x0000	0xFFFF	RW
1303	HR_CS_MQTT_BROKER_MQTT_IP_URL_16		0x0000	0xFFFF	RW
1304	HR_CS_MQTT_BROKER_MQTT_IP_URL_17		0x0000	0xFFFF	RW
1305	HR_CS_MQTT_BROKER_MQTT_IP_URL_18		0x0000	0xFFFF	RW
1306	HR_CS_MQTT_BROKER_MQTT_IP_URL_19		0x0000	0xFFFF	RW
1307	HR_CS_MQTT_BROKER_MQTT_IP_URL_20		0x0000	0xFFFF	RW
1308	HR_CS_MQTT_BROKER_MQTT_IP_URL_21		0x0000	0xFFFF	RW
1309	HR_CS_MQTT_BROKER_MQTT_IP_URL_22		0x0000	0xFFFF	RW
1310	HR_CS_MQTT_BROKER_MQTT_IP_URL_23		0x0000	0xFFFF	RW
1311	HR_CS_MQTT_BROKER_MQTT_IP_URL_24		0x0000	0xFFFF	RW
1312	HR_CS_MQTT_BROKER_MQTT_IP_URL_25		0x0000	0xFFFF	RW
1313	HR_CS_MQTT_BROKER_MQTT_IP_URL_26		0x0000	0xFFFF	RW
1314	HR_CS_MQTT_BROKER_MQTT_IP_URL_27		0x0000	0xFFFF	RW
1315	HR_CS_MQTT_BROKER_MQTT_IP_URL_28		0x0000	0xFFFF	RW
1316	HR_CS_MQTT_BROKER_MQTT_IP_URL_29		0x0000	0xFFFF	RW
1317	HR_CS_MQTT_BROKER_MQTT_IP_URL_30		0x0000	0xFFFF	RW
		Reserved.			
1319	HR_CS_MQTT_SECURITY,	Allows you to configure the protocol and data encryption for communication with the MQTT Broker: 0 → None. 1 → TLS V1.2 – Server signed. 2 → TLS V1.2 – CA only. 3 → TLS V1.2 – Self signed.	0	3	RW
1320	HR_CS_MQTT_DEVICE_ID_1	Allows you to configure an ID for the device. Each register corresponds to two characters.	0x0000	0xFFFF	RW
1321	HR_CS_MQTT_DEVICE_ID_2		0x0000	0xFFFF	RW
1322	HR_CS_MQTT_DEVICE_ID_3		0x0000	0xFFFF	RW
1323	HR_CS_MQTT_DEVICE_ID_4		0x0000	0xFFFF	RW
1324	HR_CS_MQTT_DEVICE_ID_5		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1325	HR_CS_MQTT_DEVICE_ID_6		0x0000	0xFFFF	RW
1326	HR_CS_MQTT_DEVICE_ID_7		0x0000	0xFFFF	RW
1327	HR_CS_MQTT_DEVICE_ID_8		0x0000	0xFFFF	RW
1328	HR_CS_MQTT_DEVICE_ID_9		0x0000	0xFFFF	RW
1329	HR_CS_MQTT_DEVICE_ID_10		0x0000	0xFFFF	RW
		Reserved.			
1331	HR_CS_MQTT_PRIMARY_KEY_1		0x0000	0xFFFF	RW
1332	HR_CS_MQTT_PRIMARY_KEY_2		0x0000	0xFFFF	RW
1333	HR_CS_MQTT_PRIMARY_KEY_3		0x0000	0xFFFF	RW
1334	HR_CS_MQTT_PRIMARY_KEY_4		0x0000	0xFFFF	RW
1335	HR_CS_MQTT_PRIMARY_KEY_5		0x0000	0xFFFF	RW
1336	HR_CS_MQTT_PRIMARY_KEY_6		0x0000	0xFFFF	RW
1337	HR_CS_MQTT_PRIMARY_KEY_7		0x0000	0xFFFF	RW
1338	HR_CS_MQTT_PRIMARY_KEY_8		0x0000	0xFFFF	RW
1339	HR_CS_MQTT_PRIMARY_KEY_9		0x0000	0xFFFF	RW
1340	HR_CS_MQTT_PRIMARY_KEY_10		0x0000	0xFFFF	RW
1341	HR_CS_MQTT_PRIMARY_KEY_11		0x0000	0xFFFF	RW
1342	HR_CS_MQTT_PRIMARY_KEY_12		0x0000	0xFFFF	RW
1343	HR_CS_MQTT_PRIMARY_KEY_13		0x0000	0xFFFF	RW
1344	HR_CS_MQTT_PRIMARY_KEY_14		0x0000	0xFFFF	RW
1345	HR_CS_MQTT_PRIMARY_KEY_15	Allows you to define a primary key for the device (Only for Microsoft Azure).	0x0000	0xFFFF	RW
1346	HR_CS_MQTT_PRIMARY_KEY_16		0x0000	0xFFFF	RW
1347	HR_CS_MQTT_PRIMARY_KEY_17		0x0000	0xFFFF	RW
1348	HR_CS_MQTT_PRIMARY_KEY_18		0x0000	0xFFFF	RW
1349	HR_CS_MQTT_PRIMARY_KEY_19		0x0000	0xFFFF	RW
1350	HR_CS_MQTT_PRIMARY_KEY_20		0x0000	0xFFFF	RW
1351	HR_CS_MQTT_PRIMARY_KEY_21		0x0000	0xFFFF	RW
1352	HR_CS_MQTT_PRIMARY_KEY_22		0x0000	0xFFFF	RW
1353	HR_CS_MQTT_PRIMARY_KEY_23		0x0000	0xFFFF	RW
1354	HR_CS_MQTT_PRIMARY_KEY_24		0x0000	0xFFFF	RW
1355	HR_CS_MQTT_PRIMARY_KEY_25		0x0000	0xFFFF	RW
1356	HR_CS_MQTT_PRIMARY_KEY_26		0x0000	0xFFFF	RW
1357	HR_CS_MQTT_PRIMARY_KEY_27		0x0000	0xFFFF	RW
1358	HR_CS_MQTT_PRIMARY_KEY_28		0x0000	0xFFFF	RW
1359	HR_CS_MQTT_PRIMARY_KEY_29		0x0000	0xFFFF	RW
1360	HR_CS_MQTT_PRIMARY_KEY_30		0x0000	0xFFFF	RW
		Reserved.			
1362	HR_CS_MQTT_PROJECT_ID_1		0x0000	0xFFFF	RW
1363	HR_CS_MQTT_PROJECT_ID_2		0x0000	0xFFFF	RW
1364	HR_CS_MQTT_PROJECT_ID_3		0x0000	0xFFFF	RW
1365	HR_CS_MQTT_PROJECT_ID_4		0x0000	0xFFFF	RW
1366	HR_CS_MQTT_PROJECT_ID_5	Allows you to configure an ID for the project (Only for Google Cloud).	0x0000	0xFFFF	RW
1367	HR_CS_MQTT_PROJECT_ID_6		0x0000	0xFFFF	RW
1368	HR_CS_MQTT_PROJECT_ID_7		0x0000	0xFFFF	RW
1369	HR_CS_MQTT_PROJECT_ID_8		0x0000	0xFFFF	RW
1370	HR_CS_MQTT_PROJECT_ID_9		0x0000	0xFFFF	RW
1371	HR_CS_MQTT_PROJECT_ID_10		0x0000	0xFFFF	RW
		Reserved.			

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1373	HR_CS_MQTT_GGL_REGION_1	Allows you to define a region for the connection (Only for Google Cloud).	0x0000	0xFFFF	RW
1374	HR_CS_MQTT_GGL_REGION_2		0x0000	0xFFFF	RW
1375	HR_CS_MQTT_GGL_REGION_3		0x0000	0xFFFF	RW
1376	HR_CS_MQTT_GGL_REGION_4		0x0000	0xFFFF	RW
1377	HR_CS_MQTT_GGL_REGION_5		0x0000	0xFFFF	RW
1378	HR_CS_MQTT_GGL_REGION_6		0x0000	0xFFFF	RW
1379	HR_CS_MQTT_GGL_REGION_7		0x0000	0xFFFF	RW
1380	HR_CS_MQTT_GGL_REGION_8		0x0000	0xFFFF	RW
1381	HR_CS_MQTT_GGL_REGION_9		0x0000	0xFFFF	RW
1382	HR_CS_MQTT_GGL_REGION_10		0x0000	0xFFFF	RW
1383	HR_CS_MQTT_TOPIC_RETAIN	Allows you to enable data retention in the cloud.	0	1	RW
1384	HR_CS_MQTT_TOPIC_DATA_PUB_1	Allows the device to publish data to the Channels and Events publication topics.	0x0000	0xFFFF	RW
1385	HR_CS_MQTT_TOPIC_DATA_PUB_2		0x0000	0xFFFF	RW
1386	HR_CS_MQTT_TOPIC_DATA_PUB_3		0x0000	0xFFFF	RW
1387	HR_CS_MQTT_TOPIC_DATA_PUB_4		0x0000	0xFFFF	RW
1388	HR_CS_MQTT_TOPIC_DATA_PUB_5		0x0000	0xFFFF	RW
1389	HR_CS_MQTT_TOPIC_DATA_PUB_6		0x0000	0xFFFF	RW
1390	HR_CS_MQTT_TOPIC_DATA_PUB_7		0x0000	0xFFFF	RW
1391	HR_CS_MQTT_TOPIC_DATA_PUB_8		0x0000	0xFFFF	RW
1392	HR_CS_MQTT_TOPIC_DATA_PUB_9		0x0000	0xFFFF	RW
1393	HR_CS_MQTT_TOPIC_DATA_PUB_10		0x0000	0xFFFF	RW
1394	HR_CS_MQTT_TOPIC_DATA_PUB_11		0x0000	0xFFFF	RW
1395	HR_CS_MQTT_TOPIC_DATA_PUB_12		0x0000	0xFFFF	RW
1396	HR_CS_MQTT_TOPIC_DATA_PUB_13		0x0000	0xFFFF	RW
1397	HR_CS_MQTT_TOPIC_DATA_PUB_14		0x0000	0xFFFF	RW
1398	HR_CS_MQTT_TOPIC_DATA_PUB_15		0x0000	0xFFFF	RW
1399	HR_CS_MQTT_TOPIC_DATA_PUB_16		0x0000	0xFFFF	RW
1400	HR_CS_MQTT_TOPIC_DATA_PUB_17		0x0000	0xFFFF	RW
1401	HR_CS_MQTT_TOPIC_DATA_PUB_18		0x0000	0xFFFF	RW
1402	HR_CS_MQTT_TOPIC_DATA_PUB_19		0x0000	0xFFFF	RW
1403	HR_CS_MQTT_TOPIC_DATA_PUB_20		0x0000	0xFFFF	RW
1404	HR_CS_MQTT_TOPIC_DATA_PUB_21		0x0000	0xFFFF	RW
1405	HR_CS_MQTT_TOPIC_DATA_PUB_22		0x0000	0xFFFF	RW
1406	HR_CS_MQTT_TOPIC_DATA_PUB_23		0x0000	0xFFFF	RW
1407	HR_CS_MQTT_TOPIC_DATA_PUB_24		0x0000	0xFFFF	RW
1408	HR_CS_MQTT_TOPIC_DATA_PUB_25		0x0000	0xFFFF	RW
1409	HR_CS_MQTT_TOPIC_DATA_PUB_26		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1410	HR_CS_MQTT_TOPIC_DATA_PUB_27		0x0000	0xFFFF	RW
1411	HR_CS_MQTT_TOPIC_DATA_PUB_28		0x0000	0xFFFF	RW
1412	HR_CS_MQTT_TOPIC_DATA_PUB_29		0x0000	0xFFFF	RW
1413	HR_CS_MQTT_TOPIC_DATA_PUB_30		0x0000	0xFFFF	RW
		Reserved.			
1415	HR_CS_MQTT_TOPIC_ACK_PUB_1	Allows the device to publish data to the **Ack Config** publishing topic.	0x0000	0xFFFF	RW
1416	HR_CS_MQTT_TOPIC_ACK_PUB_2		0x0000	0xFFFF	RW
1417	HR_CS_MQTT_TOPIC_ACK_PUB_3		0x0000	0xFFFF	RW
1418	HR_CS_MQTT_TOPIC_ACK_PUB_4		0x0000	0xFFFF	RW
1419	HR_CS_MQTT_TOPIC_ACK_PUB_5		0x0000	0xFFFF	RW
1420	HR_CS_MQTT_TOPIC_ACK_PUB_6		0x0000	0xFFFF	RW
1421	HR_CS_MQTT_TOPIC_ACK_PUB_7		0x0000	0xFFFF	RW
1422	HR_CS_MQTT_TOPIC_ACK_PUB_8		0x0000	0xFFFF	RW
1423	HR_CS_MQTT_TOPIC_ACK_PUB_9		0x0000	0xFFFF	RW
1424	HR_CS_MQTT_TOPIC_ACK_PUB_10		0x0000	0xFFFF	RW
1425	HR_CS_MQTT_TOPIC_ACK_PUB_11		0x0000	0xFFFF	RW
1426	HR_CS_MQTT_TOPIC_ACK_PUB_12		0x0000	0xFFFF	RW
1427	HR_CS_MQTT_TOPIC_ACK_PUB_13		0x0000	0xFFFF	RW
1428	HR_CS_MQTT_TOPIC_ACK_PUB_14		0x0000	0xFFFF	RW
1429	HR_CS_MQTT_TOPIC_ACK_PUB_15		0x0000	0xFFFF	RW
1430	HR_CS_MQTT_TOPIC_ACK_PUB_16		0x0000	0xFFFF	RW
1431	HR_CS_MQTT_TOPIC_ACK_PUB_17		0x0000	0xFFFF	RW
1432	HR_CS_MQTT_TOPIC_ACK_PUB_18		0x0000	0xFFFF	RW
1433	HR_CS_MQTT_TOPIC_ACK_PUB_19		0x0000	0xFFFF	RW
1434	HR_CS_MQTT_TOPIC_ACK_PUB_20		0x0000	0xFFFF	RW
1435	HR_CS_MQTT_TOPIC_ACK_PUB_21		0x0000	0xFFFF	RW
1436	HR_CS_MQTT_TOPIC_ACK_PUB_22		0x0000	0xFFFF	RW
1437	HR_CS_MQTT_TOPIC_ACK_PUB_23		0x0000	0xFFFF	RW
1438	HR_CS_MQTT_TOPIC_ACK_PUB_24		0x0000	0xFFFF	RW
1439	HR_CS_MQTT_TOPIC_ACK_PUB_25		0x0000	0xFFFF	RW
1440	HR_CS_MQTT_TOPIC_ACK_PUB_26		0x0000	0xFFFF	RW
1441	HR_CS_MQTT_TOPIC_ACK_PUB_27		0x0000	0xFFFF	RW
1442	HR_CS_MQTT_TOPIC_ACK_PUB_28		0x0000	0xFFFF	RW
1443	HR_CS_MQTT_TOPIC_ACK_PUB_29		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1444	HR_CS_MQTT_TOPIC_ACK_PUB_30		0x0000	0xFFFF	RW
		Reserved.			
1446	HR_CS_MQTT_TOPIC_CONFIG_SUB_1	Allows the device to receive data from the **Config** subscription topic.	0x0000	0xFFFF	RW
1447	HR_CS_MQTT_TOPIC_CONFIG_SUB_2		0x0000	0xFFFF	RW
1448	HR_CS_MQTT_TOPIC_CONFIG_SUB_3		0x0000	0xFFFF	RW
1449	HR_CS_MQTT_TOPIC_CONFIG_SUB_4		0x0000	0xFFFF	RW
1450	HR_CS_MQTT_TOPIC_CONFIG_SUB_5		0x0000	0xFFFF	RW
1451	HR_CS_MQTT_TOPIC_CONFIG_SUB_6		0x0000	0xFFFF	RW
1452	HR_CS_MQTT_TOPIC_CONFIG_SUB_7		0x0000	0xFFFF	RW
1453	HR_CS_MQTT_TOPIC_CONFIG_SUB_8		0x0000	0xFFFF	RW
1454	HR_CS_MQTT_TOPIC_CONFIG_SUB_9		0x0000	0xFFFF	RW
1455	HR_CS_MQTT_TOPIC_CONFIG_SUB_10		0x0000	0xFFFF	RW
1456	HR_CS_MQTT_TOPIC_CONFIG_SUB_11		0x0000	0xFFFF	RW
1457	HR_CS_MQTT_TOPIC_CONFIG_SUB_12		0x0000	0xFFFF	RW
1458	HR_CS_MQTT_TOPIC_CONFIG_SUB_13		0x0000	0xFFFF	RW
1459	HR_CS_MQTT_TOPIC_CONFIG_SUB_14		0x0000	0xFFFF	RW
1460	HR_CS_MQTT_TOPIC_CONFIG_SUB_15		0x0000	0xFFFF	RW
1461	HR_CS_MQTT_TOPIC_CONFIG_SUB_16		0x0000	0xFFFF	RW
1462	HR_CS_MQTT_TOPIC_CONFIG_SUB_17		0x0000	0xFFFF	RW
1463	HR_CS_MQTT_TOPIC_CONFIG_SUB_18		0x0000	0xFFFF	RW
1464	HR_CS_MQTT_TOPIC_CONFIG_SUB_19		0x0000	0xFFFF	RW
1465	HR_CS_MQTT_TOPIC_CONFIG_SUB_20		0x0000	0xFFFF	RW
1466	HR_CS_MQTT_TOPIC_CONFIG_SUB_21		0x0000	0xFFFF	RW
1467	HR_CS_MQTT_TOPIC_CONFIG_SUB_22		0x0000	0xFFFF	RW
1468	HR_CS_MQTT_TOPIC_CONFIG_SUB_23		0x0000	0xFFFF	RW
1469	HR_CS_MQTT_TOPIC_CONFIG_SUB_24		0x0000	0xFFFF	RW
1470	HR_CS_MQTT_TOPIC_CONFIG_SUB_25		0x0000	0xFFFF	RW
1471	HR_CS_MQTT_TOPIC_CONFIG_SUB_26		0x0000	0xFFFF	RW
1472	HR_CS_MQTT_TOPIC_CONFIG_SUB_27		0x0000	0xFFFF	RW
1473	HR_CS_MQTT_TOPIC_CONFIG_SUB_28		0x0000	0xFFFF	RW
1474	HR_CS_MQTT_TOPIC_CONFIG_SUB_29		0x0000	0xFFFF	RW
1475	HR_CS_MQTT_TOPIC_CONFIG_		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
	SUB_30				
		Reserved.			
1477	HR_CS_MQTT_TOPIC_CMD_SUB_1	Allows the device to receive data from the **Command** subscription topic.	0x0000	0xFFFF	RW
1478	HR_CS_MQTT_TOPIC_CMD_SUB_2		0x0000	0xFFFF	RW
1479	HR_CS_MQTT_TOPIC_CMD_SUB_3		0x0000	0xFFFF	RW
1480	HR_CS_MQTT_TOPIC_CMD_SUB_4		0x0000	0xFFFF	RW
1481	HR_CS_MQTT_TOPIC_CMD_SUB_5		0x0000	0xFFFF	RW
1482	HR_CS_MQTT_TOPIC_CMD_SUB_6		0x0000	0xFFFF	RW
1483	HR_CS_MQTT_TOPIC_CMD_SUB_7		0x0000	0xFFFF	RW
1484	HR_CS_MQTT_TOPIC_CMD_SUB_8		0x0000	0xFFFF	RW
1485	HR_CS_MQTT_TOPIC_CMD_SUB_9		0x0000	0xFFFF	RW
1486	HR_CS_MQTT_TOPIC_CMD_SUB_10		0x0000	0xFFFF	RW
1487	HR_CS_MQTT_TOPIC_CMD_SUB_11		0x0000	0xFFFF	RW
1488	HR_CS_MQTT_TOPIC_CMD_SUB_12		0x0000	0xFFFF	RW
1489	HR_CS_MQTT_TOPIC_CMD_SUB_13		0x0000	0xFFFF	RW
1490	HR_CS_MQTT_TOPIC_CMD_SUB_14		0x0000	0xFFFF	RW
1491	HR_CS_MQTT_TOPIC_CMD_SUB_15		0x0000	0xFFFF	RW
1492	HR_CS_MQTT_TOPIC_CMD_SUB_16		0x0000	0xFFFF	RW
1493	HR_CS_MQTT_TOPIC_CMD_SUB_17		0x0000	0xFFFF	RW
1494	HR_CS_MQTT_TOPIC_CMD_SUB_18		0x0000	0xFFFF	RW
1495	HR_CS_MQTT_TOPIC_CMD_SUB_19		0x0000	0xFFFF	RW
1496	HR_CS_MQTT_TOPIC_CMD_SUB_20		0x0000	0xFFFF	RW
1497	HR_CS_MQTT_TOPIC_CMD_SUB_21		0x0000	0xFFFF	RW
1498	HR_CS_MQTT_TOPIC_CMD_SUB_22		0x0000	0xFFFF	RW
1499	HR_CS_MQTT_TOPIC_CMD_SUB_23		0x0000	0xFFFF	RW
1500	HR_CS_MQTT_TOPIC_CMD_SUB_24		0x0000	0xFFFF	RW
1501	HR_CS_MQTT_TOPIC_CMD_SUB_25		0x0000	0xFFFF	RW
1502	HR_CS_MQTT_TOPIC_CMD_SUB_26		0x0000	0xFFFF	RW
1503	HR_CS_MQTT_TOPIC_CMD_SUB_27		0x0000	0xFFFF	RW
1504	HR_CS_MQTT_TOPIC_CMD_SUB_28		0x0000	0xFFFF	RW
1505	HR_CS_MQTT_TOPIC_CMD_SUB_29		0x0000	0xFFFF	RW
1506	HR_CS_MQTT_TOPIC_CMD_SUB_30		0x0000	0xFFFF	RW
		Reserved.			
1508	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_1	Allows the device to publish data to the **Ack Command** publication topic.	0x0000	0xFFFF	RW
1509	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_2		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1510	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_3		0x0000	0xFFFF	RW
1511	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_4		0x0000	0xFFFF	RW
1512	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_5		0x0000	0xFFFF	RW
1513	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_6		0x0000	0xFFFF	RW
1514	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_7		0x0000	0xFFFF	RW
1515	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_8		0x0000	0xFFFF	RW
1516	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_9		0x0000	0xFFFF	RW
1517	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_10		0x0000	0xFFFF	RW
1518	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_11		0x0000	0xFFFF	RW
1519	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_12		0x0000	0xFFFF	RW
1520	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_13		0x0000	0xFFFF	RW
1521	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_14		0x0000	0xFFFF	RW
1522	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_15		0x0000	0xFFFF	RW
1523	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_16		0x0000	0xFFFF	RW
1524	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_17		0x0000	0xFFFF	RW
1525	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_18		0x0000	0xFFFF	RW
1526	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_19		0x0000	0xFFFF	RW
1527	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_20		0x0000	0xFFFF	RW
1528	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_21		0x0000	0xFFFF	RW
1529	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_22		0x0000	0xFFFF	RW
1530	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_23		0x0000	0xFFFF	RW
1531	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_24		0x0000	0xFFFF	RW
1532	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_25		0x0000	0xFFFF	RW
1533	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_26		0x0000	0xFFFF	RW
1534	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_27		0x0000	0xFFFF	RW
1535	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_28		0x0000	0xFFFF	RW
1536	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_29		0x0000	0xFFFF	RW
1537	HR_CS_MQTT_TOPIC_CMD_ACK_PUB_30		0x0000	0xFFFF	RW
		Reserved.			
1539	HR_CS_MQTT_REGISTER_ID_1	Allows you to configure an ID for the register (Only for Google Cloud).	0x0000	0xFFFF	RW
1540	HR_CS_MQTT_REGISTER_ID_2		0x0000	0xFFFF	RW
1541	HR_CS_MQTT_REGISTER_ID_3		0x0000	0xFFFF	RW
1542	HR_CS_MQTT_REGISTER_ID_4		0x0000	0xFFFF	RW
1543	HR_CS_MQTT_REGISTER_ID_5		0x0000	0xFFFF	RW
1544	HR_CS_MQTT_REGISTER_ID_6		0x0000	0xFFFF	RW
1545	HR_CS_MQTT_REGISTER_ID_7		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1546	HR_CS_MQTT_REGISTER_ID_8		0x0000	0xFFFF	RW
1547	HR_CS_MQTT_REGISTER_ID_9		0x0000	0xFFFF	RW
1548	HR_CS_MQTT_REGISTER_ID_10		0x0000	0xFFFF	RW
1549	HR_CS_MQTT_SEND_DIAGCOUNTERS	Adds system counters in MQTT diagnostics and enables periodic diagnostic publishing in the command confirmation topic.	0	1	RW
1550	HR_CS_MQTT_CLOUD_SEL	Allows you to configure the cloud to be used by the MQTT protocol: 0 → Generic cloud. 1 → Google Cloud. 2 → Amazon AWS. 3 → Microsoft Azure. 4 → NOVUS Cloud.	0	4	RW
1551	HR_CS_MQTT_TOKEN_EXP	Allows you to configure the token expiration time (in minutes). Valid for connections to Google Cloud and Amazon AWS clouds.	0	65535	RW
		Reserved.			
1553	HR_CS_SNTP_SERVER_ENABLE	Allows you to enable the NTP server.	0	1	RW
1554	HR_CS_SNTP_SERVER_MIN_DIFF	Allows you to configure the minimum difference between the **DigiRail IoT** clock and the information received via the NTP server to update (in seconds).	1	0xFFFF	RW
1555	HR_CS_SNTP_SERVER_IP_URL_1	Allows you to define the IP or URL of the NTP server.	0x0000	0xFFFF	RW
1556	HR_CS_SNTP_SERVER_IP_URL_2		0x0000	0xFFFF	RW
1557	HR_CS_SNTP_SERVER_IP_URL_3		0x0000	0xFFFF	RW
1558	HR_CS_SNTP_SERVER_IP_URL_4		0x0000	0xFFFF	RW
1559	HR_CS_SNTP_SERVER_IP_URL_5		0x0000	0xFFFF	RW
1560	HR_CS_SNTP_SERVER_IP_URL_6		0x0000	0xFFFF	RW
1561	HR_CS_SNTP_SERVER_IP_URL_7		0x0000	0xFFFF	RW
1562	HR_CS_SNTP_SERVER_IP_URL_8		0x0000	0xFFFF	RW
1563	HR_CS_SNTP_SERVER_IP_URL_9		0x0000	0xFFFF	RW
1564	HR_CS_SNTP_SERVER_IP_URL_10		0x0000	0xFFFF	RW
1565	HR_CS_SNTP_SERVER_IP_URL_11		0x0000	0xFFFF	RW
1566	HR_CS_SNTP_SERVER_IP_URL_12		0x0000	0xFFFF	RW
1567	HR_CS_SNTP_SERVER_IP_URL_13		0x0000	0xFFFF	RW
1568	HR_CS_SNTP_SERVER_IP_URL_14		0x0000	0xFFFF	RW
1569	HR_CS_SNTP_SERVER_IP_URL_15		0x0000	0xFFFF	RW
1570	HR_CS_SNTP_SERVER_IP_URL_16		0x0000	0xFFFF	RW
1571	HR_CS_SNTP_SERVER_IP_URL_17		0x0000	0xFFFF	RW
1572	HR_CS_SNTP_SERVER_IP_URL_18		0x0000	0xFFFF	RW
1573	HR_CS_SNTP_SERVER_IP_URL_19		0x0000	0xFFFF	RW
1574	HR_CS_SNTP_SERVER_IP_URL_20		0x0000	0xFFFF	RW
1575	HR_CS_SNTP_SERVER_IP_URL_21		0x0000	0xFFFF	RW
1576	HR_CS_SNTP_SERVER_IP_URL_22		0x0000	0xFFFF	RW
1577	HR_CS_SNTP_SERVER_IP_URL_23		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1578	HR_CS_SNTP_SERVER_IP_URL_24		0x0000	0xFFFF	RW
1579	HR_CS_SNTP_SERVER_IP_URL_25		0x0000	0xFFFF	RW
1580	HR_CS_SNTP_SERVER_IP_URL_26		0x0000	0xFFFF	RW
1581	HR_CS_SNTP_SERVER_IP_URL_27		0x0000	0xFFFF	RW
1582	HR_CS_SNTP_SERVER_IP_URL_28		0x0000	0xFFFF	RW
1583	HR_CS_SNTP_SERVER_IP_URL_29		0x0000	0xFFFF	RW
1584	HR_CS_SNTP_SERVER_IP_URL_30		0x0000	0xFFFF	RW
		Reserved.			
1621	HR_CS_MODBUS_WDT_INTERVAL	Modbus-TCP Watchdog interval in seconds.	0	300	RW
1622	HR_CS_MODBUS_WDT_ENABLE	Enables Modbus-TCP Watchdog operation with the value (0xA5).	0x0000	0xFFFF	RW
		Reserved.			
1624	HR_CS_RS485_OPERMODE	RS485 operation mode: 0 → Disabled. 1 → Gateway. 2 → Master. 3 → Slave.	0	3	RW
1625	HR_CS_MODBUS_RTU_ADDRESS	Modbus-RTU address to which the device will respond when operating in slave mode.	1	247	RW
1626	HR_CS_RS485_NUMBER_OF_READ_ATTEMPTS	Number of read attempts to be sent to the slave device.	1	65535	RW
		Reserved.			
1631	HR_CS_MODBUS_MASTER_CH1_ENABLED	Enable remote channel 1.	0	1	RW
1632	HR_CS_MODBUS_MASTER_CH1_SLAVE_ADDR	Address of the slave device linked to the channel.	1	247	RW
1633	HR_CS_MODBUS_MASTER_CH1_REGISTER_ADDR	Address of the register linked to the virtual channel.	0	65535	RW
1634	HR_CS_MODBUS_MASTER_CH1_COMMAND_MODBUS	Modbus read command to be sent: 0 → 0x02 Command (Discrete Input Register). 1 → 0x03 Command (Holding Register). 2 → 0x04 Command (Input Registers).	0	2	RW
1635	HR_CS_MODBUS_MASTER_CH1_DATA_TYPE	Type of data to be read: 0 → 16-bits unsigned. 1 →16-bits signed. 2 → 32-bits unsigned. 3 → 32-bits signed. 4 → 64-bits unsigned. 5 → 64-bits signed. 6 → 32-bits float (Single Precision Floating Point). 7 → 64-bits double (Double Precision Floating Point).	0	7	RW
1636	HR_CS_MODBUS_MASTER_CH1_DECIMAL_POINT	Number of decimal places in the read data.	0	15	RW
1637	HR_CS_MODBUS_MASTER_CH1_BYTES_INVERSION	Orientation of the bytes in the read data: 0 → No inversion. 1 → Byte inversion. 2 → Word inversion (16-bits) (Short). 3 → Word and byte inversion. 4 → Double-word inversion (32-bits) (Long). 5 → Double-word and byte inversion. 6 → Long and Short inversion. 7 → Total inversion (Byte, Word and Double-word).	0	7	RW
1638	HR_CS_MODBUS_MASTER_CH1_ERROR_PART_HH	Register for setting the error value linked to the virtual channel.	0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1639	HR_CS_MODBUS_MASTER_CH1_ ERROR_PART_HL	Register _LL error value set for the channel when it is configured to read 16 bits of data. Registers _LH _LL high and low part of the error value when the channel is set to 32-bits. Registers _HH to _LL correspond to the 64-bit error value when the channel is configured like this.	0x0000	0xFFFF	RW
1640	HR_CS_MODBUS_MASTER_CH1_ ERROR_PART_LH		0x0000	0xFFFF	RW
1641	HR_CS_MODBUS_MASTER_CH1_ ERROR_PART_LL		0x0000	0xFFFF	RW
1642	HR_CS_MODBUS_MASTER_CH1_ ERROR_ENABLED	Enable the display of the error value linked to the channel in the event of a communication failure.	0	1	RW
		Reserved.			
1644	HR_CS_MODBUS_MASTER_CH2_ ENABLED	Enable remote channel 2.	0	1	RW
1645	HR_CS_MODBUS_MASTER_CH2_ SLAVE_ADDR	Address of the slave device linked to the channel.	1	247	RW
1646	HR_CS_MODBUS_MASTER_CH2_ REGISTER_ADDR	Address of the register linked to the virtual channel.	0	65535	RW
1647	HR_CS_MODBUS_MASTER_CH2_ COMMAND_MODBUS	Modbus read command to be sent: 0 → 0x02 Command (Discrete Input Register). 1 → 0x03 Command (Holding Register). 2 → 0x04 Command (Input Registers).	0	2	RW
1648	HR_CS_MODBUS_MASTER_CH2_ DATA_TYPE	Type of data to be read: 0 → 16-bits unsigned. 1 → 16-bits signed. 2 → 32-bits unsigned. 3 → 32-bits signed. 4 → 64-bits unsigned. 5 → 64-bits signed. 6 → 32-bits float (Single Precision Floating Point). 7 → 64-bits double (Double Precision Floating Point).	0	7	RW
1649	HR_CS_MODBUS_MASTER_CH2_ DECIMAL_POINT	Number of decimal places in the read data.	0	15	RW
1650	HR_CS_MODBUS_MASTER_CH2_ BYTES_INVERSION	Orientation of the bytes in the read data: 0 → No inversion. 1 → Byte inversion. 2 → Word inversion (16-bits) (Short). 3 → Word and byte inversion. 4 → Double-word inversion (32-bits) (Long). 5 → Double-word and byte inversion. 6 → Long and Short inversion. 7 → Total inversion (Byte, Word and Double-word).	0	7	RW
1651	HR_CS_MODBUS_MASTER_CH2_ ERROR_PART_HH	Register for setting the error value linked to the virtual channel.	0x0000	0xFFFF	RW
1652	HR_CS_MODBUS_MASTER_CH2_ ERROR_PART_HL	Register _LL error value set for the channel when it is configured to read 16 bits of data. Registers _LH _LL high and low part of the error value when the channel is set to 32-bits. Registers _HH to _LL correspond to the 64-bit error value when the channel is configured like this.	0x0000	0xFFFF	RW
1653	HR_CS_MODBUS_MASTER_CH2_ ERROR_PART_LH		0x0000	0xFFFF	RW
1654	HR_CS_MODBUS_MASTER_CH2_ ERROR_PART_LL		0x0000	0xFFFF	RW
1655	HR_CS_MODBUS_MASTER_CH2_ ERROR_ENABLED	Enable the display of the error value linked to the channel in the event of a communication failure.	0	1	RW
		Reserved.			
1657	HR_CS_MODBUS_MASTER_CH3_ ENABLED	Enable remote channel 3.	0	1	RW
1658	HR_CS_MODBUS_MASTER_CH3_ SLAVE_ADDR	Address of the slave device linked to the channel.	1	247	RW
1659	HR_CS_MODBUS_MASTER_CH3_ REGISTER_ADDR	Address of the register linked to the virtual channel.	0	65535	RW
1660	HR_CS_MODBUS_MASTER_CH3_ COMMAND_MODBUS	Modbus read command to be sent: 0 → 0x02 Command (Discrete Input Register). 1 → 0x03 Command (Holding Register). 2 → 0x04 Command (Input Registers).	0	2	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1661	HR_CS_MODBUS_MASTER_CH3_ DATA_TYPE	Type of data to be read: 0 → 16-bits unsigned. 1 →16-bits signed. 2 → 32-bits unsigned. 3 → 32-bits signed. 4 → 64-bits unsigned. 5 → 64-bits signed. 6 → 32-bits float (Single Precision Floating Point). 7 → 64-bits double (Double Precision Floating Point).	0	7	RW
1662	HR_CS_MODBUS_MASTER_CH3_ DECIMAL_POINT	Number of decimal places in the read data.	0	15	RW
1663	HR_CS_MODBUS_MASTER_CH3_ BYTES_INVERSION	Orientation of the bytes in the read data: 0 → No inversion. 1 → Byte inversion. 2 → Word inversion (16-bits) (Short). 3 → Word and byte inversion. 4 → Double-word inversion (32-bits) (Long). 5 → Double-word and byte inversion. 6 → Long and Short inversion. 7 → Total inversion (Byte, Word and Double-word).	0	7	RW
1664	HR_CS_MODBUS_MASTER_CH3_ ERROR_PART_HH	Register for setting the error value linked to the virtual channel. Register _LL error value set for the channel when it is configured to read 16 bits of data. Registers _LH _LL high and low part of the error value when the channel is set to 32-bits. Registers _HH to _LL correspond to the 64-bit error value when the channel is configured like this.	0x0000	0xFFFF	RW
1665	HR_CS_MODBUS_MASTER_CH3_ ERROR_PART_HL		0x0000	0xFFFF	RW
1666	HR_CS_MODBUS_MASTER_CH3_ ERROR_PART_LH		0x0000	0xFFFF	RW
1667	HR_CS_MODBUS_MASTER_CH3_ ERROR_PART_LL		0x0000	0xFFFF	RW
1668	HR_CS_MODBUS_MASTER_CH3_ ERROR_ENABLED	Enable the display of the error value linked to the channel in the event of a communication failure.	0	1	RW
		Reserved.			
1670	HR_CS_MODBUS_MASTER_CH4_ ENABLED	Enable remote channel 4.	0	1	RW
1671	HR_CS_MODBUS_MASTER_CH4_ SLAVE_ADDR	Address of the slave device linked to the channel.	1	247	RW
1672	HR_CS_MODBUS_MASTER_CH4_ REGISTER_ADDR	Address of the register linked to the virtual channel.	0	65535	RW
1673	HR_CS_MODBUS_MASTER_CH4_ COMMAND_MODBUS	Modbus read command to be sent: 0 → 0x02 Command (Discrete Input Register). 1 → 0x03 Command (Holding Register). 2 → 0x04 Command (Input Registers).	0	2	RW
1674	HR_CS_MODBUS_MASTER_CH4_ DATA_TYPE	Type of data to be read: 0 → 16-bits unsigned. 1 →16-bits signed. 2 → 32-bits unsigned. 3 → 32-bits signed. 4 → 64-bits unsigned. 5 → 64-bits signed. 6 → 32-bits float (Single Precision Floating Point). 7 → 64-bits double (Double Precision Floating Point).	0	7	RW
1675	HR_CS_MODBUS_MASTER_CH4_ DECIMAL_POINT	Number of decimal places in the read data.	0	15	RW
1676	HR_CS_MODBUS_MASTER_CH4_ BYTES_INVERSION	Orientation of the bytes in the read data: 0 → No inversion. 1 → Byte inversion. 2 → Word inversion (16-bits) (Short). 3 → Word and byte inversion. 4 → Double-word inversion (32-bits) (Long). 5 → Double-word and byte inversion.	0	7	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
		6 → Long and Short inversion. 7 → Total inversion (Byte, Word and Double-word).			
1677	HR_CS_MODBUS_MASTER_CH4_ ERROR_PART_HH	Register for setting the error value linked to the virtual channel. Register _LL error value set for the channel when it is configured to read 16 bits of data. Registers _LH _LL high and low part of the error value when the channel is set to 32-bits. Registers _HH to _LL correspond to the 64-bit error value when the channel is configured like this.	0x0000	0xFFFF	RW
1678	HR_CS_MODBUS_MASTER_CH4_ ERROR_PART_HL		0x0000	0xFFFF	RW
1679	HR_CS_MODBUS_MASTER_CH4_ ERROR_PART_LH		0x0000	0xFFFF	RW
1680	HR_CS_MODBUS_MASTER_CH4_ ERROR_PART_LL		0x0000	0xFFFF	RW
1681	HR_CS_MODBUS_MASTER_CH4_ ERROR_ENABLED	Enable the display of the error value linked to the channel in the event of a communication failure.	0	1	RW
		Reserved.			
1683	HR_CS_MODBUS_MASTER_CH5_ ENABLED	Enable remote channel 5.	0	1	RW
1684	HR_CS_MODBUS_MASTER_CH5_ SLAVE_ADDR	Address of the slave device linked to the channel.	1	247	RW
1685	HR_CS_MODBUS_MASTER_CH5_ REGISTER_ADDR	Address of the register linked to the virtual channel.	0	65535	RW
1686	HR_CS_MODBUS_MASTER_CH5_ COMMAND_MODBUS	Modbus read command to be sent: 0 → 0x02 Command (Discrete Input Register). 1 → 0x03 Command (Holding Register). 2 → 0x04 Command (Input Registers).	0	2	RW
1687	HR_CS_MODBUS_MASTER_CH5_ DATA_TYPE	Type of data to be read: 0 → 16-bits unsigned. 1 →16-bits signed. 2 → 32-bits unsigned. 3 → 32-bits signed. 4 → 64-bits unsigned. 5 → 64-bits signed. 6 → 32-bits float (Single Precision Floating Point). 7 → 64-bits double (Double Precision Floating Point).	0	7	RW
1688	HR_CS_MODBUS_MASTER_CH5_ DECIMAL_POINT	Number of decimal places in the read data.	0	15	RW
1689	HR_CS_MODBUS_MASTER_CH5_ BYTES_INVERSION	Orientation of the bytes in the read data: 0 → No inversion. 1 → Byte inversion. 2 → Word inversion (16-bits) (Short). 3 → Word and byte inversion. 4 → Double-word inversion (32-bits) (Long). 5 → Double-word and byte inversion. 6 → Long and Short inversion. 7 → Total inversion (Byte, Word and Double-word).	0	7	RW
1690	HR_CS_MODBUS_MASTER_CH5_ ERROR_PART_HH	Register for setting the error value linked to the virtual channel. Register _LL error value set for the channel when it is configured to read 16 bits of data. Registers _LH _LL high and low part of the error value when the channel is set to 32-bits. Registers _HH to _LL correspond to the 64-bit error value when the channel is configured like this.	0x0000	0xFFFF	RW
1691	HR_CS_MODBUS_MASTER_CH5_ ERROR_PART_HL		0x0000	0xFFFF	RW
1692	HR_CS_MODBUS_MASTER_CH5_ ERROR_PART_LH		0x0000	0xFFFF	RW
1693	HR_CS_MODBUS_MASTER_CH5_ ERROR_PART_LL		0x0000	0xFFFF	RW
1694	HR_CS_MODBUS_MASTER_CH5_ ERROR_ENABLED	Enable the display of the error value linked to the channel in the event of a communication failure.	0	1	RW
		Reserved.			
1696	HR_CS_MODBUS_MASTER_CH6_ ENABLED	Enable remote channel 7.	0	1	RW
1697	HR_CS_MODBUS_MASTER_CH6_ SLAVE_ADDR	Address of the slave device linked to the channel.	1	247	RW
1698	HR_CS_MODBUS_MASTER_CH6_ REGISTER_ADDR	Address of the register linked to the virtual channel.	0	65535	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
1699	HR_CS_MODBUS_MASTER_CH6_ COMMAND_MODBUS	Modbus read command to be sent: 0 → 0x02 Command (Discrete Input Register). 1 → 0x03 Command (Holding Register). 2 → 0x04 Command (Input Registers).	0	2	RW
1700	HR_CS_MODBUS_MASTER_CH6_ DATA_TYPE	Type of data to be read: 0 → 16-bits unsigned. 1 →16-bits signed. 2 → 32-bits unsigned. 3 → 32-bits signed. 4 → 64-bits unsigned. 5 → 64-bits signed. 6 → 32-bits float (Single Precision Floating Point). 7 → 64-bits double (Double Precision Floating Point).	0	7	RW
1701	HR_CS_MODBUS_MASTER_CH6_ DECIMAL_POINT	Number of decimal places in the read data.	0	15	RW
1702	HR_CS_MODBUS_MASTER_CH6_ BYTES_INVERSION	Orientation of the bytes in the read data: 0 → No inversion. 1 → Byte inversion. 2 → Word inversion (16-bits) (Short). 3 → Word and byte inversion. 4 → Double-word inversion (32-bits) (Long). 5 → Double-word and byte inversion. 6 → Long and Short inversion. 7 → Total inversion (Byte, Word and Double-word).	0	7	RW
1703	HR_CS_MODBUS_MASTER_CH6_ ERROR_PART_HH	Register for setting the error value linked to the virtual channel. Register _LL error value set for the channel when it is configured to read 16 bits of data. Registers _LH _LL high and low part of the error value when the channel is set to 32-bits. Registers _HH to _LL correspond to the 64-bit error value when the channel is configured like this.	0x0000	0xFFFF	RW
1704	HR_CS_MODBUS_MASTER_CH6_ ERROR_PART_HL		0x0000	0xFFFF	RW
1705	HR_CS_MODBUS_MASTER_CH6_ ERROR_PART_LH		0x0000	0xFFFF	RW
1706	HR_CS_MODBUS_MASTER_CH6_ ERROR_PART_LL		0x0000	0xFFFF	RW
1707	HR_CS_MODBUS_MASTER_CH6_ ERROR_ENABLED	Enable the display of the error value linked to the channel in the event of a communication failure.	0	1	RW
		Reserved.			
1709	HR_CS_MODBUS_MASTER_CH7_ ENABLED	Enable remote channel 7.	0	1	RW
1710	HR_CS_MODBUS_MASTER_CH7_ SLAVE_ADDR	Address of the slave device linked to the channel.	1	247	RW
1711	HR_CS_MODBUS_MASTER_CH7_ REGISTER_ADDR	Address of the register linked to the virtual channel.	0	65535	RW
1712	HR_CS_MODBUS_MASTER_CH7_ COMMAND_MODBUS	Modbus read command to be sent: 0 → 0x02 Command (Discrete Input Register). 1 → 0x03 Command (Holding Register). 2 → 0x04 Command (Input Registers).	0	2	RW
1713	HR_CS_MODBUS_MASTER_CH7_ DATA_TYPE	Type of data to be read: 0 → 16-bits unsigned. 1 →16-bits signed. 2 → 32-bits unsigned. 3 → 32-bits signed. 4 → 64-bits unsigned. 5 → 64-bits signed. 6 → 32-bits float (Single Precision Floating Point). 7 → 64-bits double (Double Precision Floating Point).	0	7	RW
1714	HR_CS_MODBUS_MASTER_CH7_ DECIMAL_POINT	Number of decimal places in the read data.	0	15	RW
1715	HR_CS_MODBUS_MASTER_CH7_ BYTES_INVERSION	Orientation of the bytes in the read data: 0 → No inversion. 1 → Byte inversion.	0	7	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
		2 → Word inversion (16-bits) (Short). 3 → Word and byte inversion. 4 → Double-word inversion (32-bits) (Long). 5 → Double-word and byte inversion. 6 → Long and Short inversion. 7 → Total inversion (Byte, Word and Double-word).			
1716	HR_CS_MODBUS_MASTER_CH7_ ERROR_PART_HH	Register for setting the error value linked to the virtual channel.	0x0000	0xFFFF	RW
1717	HR_CS_MODBUS_MASTER_CH7_ ERROR_PART_HL	Register _LL error value set for the channel when it is configured to read 16 bits of data.	0x0000	0xFFFF	RW
1718	HR_CS_MODBUS_MASTER_CH7_ ERROR_PART_LH	Registers _LH _LL high and low part of the error value when the channel is set to 32-bits.	0x0000	0xFFFF	RW
1719	HR_CS_MODBUS_MASTER_CH7_ ERROR_PART_LL	Registers _HH to _LL correspond to the 64-bit error value when the channel is configured like this.	0x0000	0xFFFF	RW
1720	HR_CS_MODBUS_MASTER_CH7_ ERROR_ENABLED	Enable the display of the error value linked to the channel in the event of a communication failure.	0	1	RW
		Reserved.			
1722	HR_CS_MODBUS_MASTER_CH8_ ENABLED	Enable remote channel 8.	0	1	RW
1723	HR_CS_MODBUS_MASTER_CH8_ SLAVE_ADDR	Address of the slave device linked to the channel.	1	247	RW
1724	HR_CS_MODBUS_MASTER_CH8_ REGISTER_ADDR	Address of the register linked to the virtual channel.	0	65535	RW
1725	HR_CS_MODBUS_MASTER_CH8_ COMMAND_MODBUS	Modbus read command to be sent: 0 → 0x02 Command (Discrete Input Register). 1 → 0x03 Command (Holding Register). 2 → 0x04 Command (Input Registers).	0	2	RW
1726	HR_CS_MODBUS_MASTER_CH8_ DATA_TYPE	Type of data to be read: 0 → 16-bits unsigned. 1 →16-bits signed. 2 → 32-bits unsigned. 3 → 32-bits signed. 4 → 64-bits unsigned. 5 → 64-bits signed. 6 → 32-bits float (Single Precision Floating Point). 7 → 64-bits double (Double Precision Floating Point).	0	7	RW
1727	HR_CS_MODBUS_MASTER_CH8_ DECIMAL_POINT	Number of decimal places in the read data.	0	15	RW
1728	HR_CS_MODBUS_MASTER_CH8_ BYTES_INVERSION	Orientation of the bytes in the read data: 0 → No inversion. 1 → Byte inversion. 2 → Word inversion (16-bits) (Short). 3 → Word and byte inversion. 4 → Double-word inversion (32-bits) (Long). 5 → Double-word and byte inversion. 6 → Long and Short inversion. 7 → Total inversion (Byte, Word and Double-word).	0	7	RW
1729	HR_CS_MODBUS_MASTER_CH8_ ERROR_PART_HH	Register for setting the error value linked to the virtual channel.	0x0000	0xFFFF	RW
1730	HR_CS_MODBUS_MASTER_CH8_ ERROR_PART_HL	Register _LL error value set for the channel when it is configured to read 16 bits of data.	0x0000	0xFFFF	RW
1731	HR_CS_MODBUS_MASTER_CH8_ ERROR_PART_LH	Registers _LH _LL high and low part of the error value when the channel is set to 32-bits.	0x0000	0xFFFF	RW
1732	HR_CS_MODBUS_MASTER_CH8_ ERROR_PART_LL	Registers _HH to _LL correspond to the 64-bit error value when the channel is configured like this.	0x0000	0xFFFF	RW
1733	HR_CS_MODBUS_MASTER_CH8_ ERROR_ENABLED	Enable the display of the error value linked to the channel in the event of a communication failure.	0	1	RW
		Reserved.			
1740	HR_CS_MQTT_BROKER_LONG_	Allows you to define the Broker user (64	0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
	USER_1	characters). Each register is equivalent to 2 characters.			
1741	HR_CS_MQTT_BROKER_LONG_ USER_2		0x0000	0xFFFF	RW
1742	HR_CS_MQTT_BROKER_LONG_ USER_3		0x0000	0xFFFF	RW
1743	HR_CS_MQTT_BROKER_LONG_ USER_4		0x0000	0xFFFF	RW
1744	HR_CS_MQTT_BROKER_LONG_ USER_5		0x0000	0xFFFF	RW
1745	HR_CS_MQTT_BROKER_LONG_ USER_6		0x0000	0xFFFF	RW
1746	HR_CS_MQTT_BROKER_LONG_ USER_7		0x0000	0xFFFF	RW
1747	HR_CS_MQTT_BROKER_LONG_ USER_8		0x0000	0xFFFF	RW
1748	HR_CS_MQTT_BROKER_LONG_ USER_9		0x0000	0xFFFF	RW
1749	HR_CS_MQTT_BROKER_LONG_ USER_10		0x0000	0xFFFF	RW
1750	HR_CS_MQTT_BROKER_LONG_ USER_11		0x0000	0xFFFF	RW
1751	HR_CS_MQTT_BROKER_LONG_ USER_12		0x0000	0xFFFF	RW
1752	HR_CS_MQTT_BROKER_LONG_ USER_13		0x0000	0xFFFF	RW
1753	HR_CS_MQTT_BROKER_LONG_ USER_14		0x0000	0xFFFF	RW
1754	HR_CS_MQTT_BROKER_LONG_ USER_15		0x0000	0xFFFF	RW
1755	HR_CS_MQTT_BROKER_LONG_ USER_16		0x0000	0xFFFF	RW
1756	HR_CS_MQTT_BROKER_LONG_ USER_17		0x0000	0xFFFF	RW
1757	HR_CS_MQTT_BROKER_LONG_ USER_18		0x0000	0xFFFF	RW
1758	HR_CS_MQTT_BROKER_LONG_ USER_19		0x0000	0xFFFF	RW
1759	HR_CS_MQTT_BROKER_LONG_ USER_20		0x0000	0xFFFF	RW
1760	HR_CS_MQTT_BROKER_LONG_ USER_21		0x0000	0xFFFF	RW
1761	HR_CS_MQTT_BROKER_LONG_ USER_22		0x0000	0xFFFF	RW
1762	HR_CS_MQTT_BROKER_LONG_ USER_23		0x0000	0xFFFF	RW
1763	HR_CS_MQTT_BROKER_LONG_ USER_24		0x0000	0xFFFF	RW
1764	HR_CS_MQTT_BROKER_LONG_ USER_25		0x0000	0xFFFF	RW
1765	HR_CS_MQTT_BROKER_LONG_ USER_26		0x0000	0xFFFF	RW
1766	HR_CS_MQTT_BROKER_LONG_ USER_27		0x0000	0xFFFF	RW
1767	HR_CS_MQTT_BROKER_LONG_ USER_28		0x0000	0xFFFF	RW
1768	HR_CS_MQTT_BROKER_LONG_ USER_29		0x0000	0xFFFF	RW
1769	HR_CS_MQTT_BROKER_LONG_ USER_30		0x0000	0xFFFF	RW
1770	HR_CS_MQTT_BROKER_LONG_ USER_31		0x0000	0xFFFF	RW
1771	HR_CS_MQTT_BROKER_LONG_ USER_32		0x0000	0xFFFF	RW
		Reserved.			
1780	HR_CS_MQTT_BROKER_LONG_ PASSWORD_1	Allows you to set a password for the Broker (64 characters). Each register is equivalent to 2 characters.	0x0000	0xFFFF	RW
1781	HR_CS_MQTT_BROKER_LONG_		0x0000	0xFFFF	RW

ADDRESS	REGISTER	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE	TYPE
	PASSWORD_2				
1782	HR_CS_MQTT_BROKER_LONG_ PASSWORD_3		0x0000	0xFFFF	RW
1783	HR_CS_MQTT_BROKER_LONG_ PASSWORD_4		0x0000	0xFFFF	RW
1784	HR_CS_MQTT_BROKER_LONG_ PASSWORD_5		0x0000	0xFFFF	RW
1785	HR_CS_MQTT_BROKER_LONG_ PASSWORD_6		0x0000	0xFFFF	RW
1786	HR_CS_MQTT_BROKER_LONG_ PASSWORD_7		0x0000	0xFFFF	RW
1787	HR_CS_MQTT_BROKER_LONG_ PASSWORD_8		0x0000	0xFFFF	RW
1788	HR_CS_MQTT_BROKER_LONG_ PASSWORD_9		0x0000	0xFFFF	RW
1789	HR_CS_MQTT_BROKER_LONG_ PASSWORD_10		0x0000	0xFFFF	RW
1790	HR_CS_MQTT_BROKER_LONG_ PASSWORD_11		0x0000	0xFFFF	RW
1791	HR_CS_MQTT_BROKER_LONG_ PASSWORD_12		0x0000	0xFFFF	RW
1792	HR_CS_MQTT_BROKER_LONG_ PASSWORD_13		0x0000	0xFFFF	RW
1793	HR_CS_MQTT_BROKER_LONG_ PASSWORD_14		0x0000	0xFFFF	RW
1794	HR_CS_MQTT_BROKER_LONG_ PASSWORD_15		0x0000	0xFFFF	RW
1795	HR_CS_MQTT_BROKER_LONG_ PASSWORD_16		0x0000	0xFFFF	RW
1796	HR_CS_MQTT_BROKER_LONG_ PASSWORD_17		0x0000	0xFFFF	RW
1797	HR_CS_MQTT_BROKER_LONG_ PASSWORD_18		0x0000	0xFFFF	RW
1798	HR_CS_MQTT_BROKER_LONG_ PASSWORD_19		0x0000	0xFFFF	RW
1799	HR_CS_MQTT_BROKER_LONG_ PASSWORD_20		0x0000	0xFFFF	RW
1800	HR_CS_MQTT_BROKER_LONG_ PASSWORD_21		0x0000	0xFFFF	RW
1801	HR_CS_MQTT_BROKER_LONG_ PASSWORD_22		0x0000	0xFFFF	RW
1802	HR_CS_MQTT_BROKER_LONG_ PASSWORD_23		0x0000	0xFFFF	RW
1803	HR_CS_MQTT_BROKER_LONG_ PASSWORD_24		0x0000	0xFFFF	RW
1804	HR_CS_MQTT_BROKER_LONG_ PASSWORD_25		0x0000	0xFFFF	RW
1805	HR_CS_MQTT_BROKER_LONG_ PASSWORD_26		0x0000	0xFFFF	RW
1806	HR_CS_MQTT_BROKER_LONG_ PASSWORD_27		0x0000	0xFFFF	RW
1807	HR_CS_MQTT_BROKER_LONG_ PASSWORD_28		0x0000	0xFFFF	RW
1808	HR_CS_MQTT_BROKER_LONG_ PASSWORD_29		0x0000	0xFFFF	RW
1809	HR_CS_MQTT_BROKER_LONG_ PASSWORD_30		0x0000	0xFFFF	RW
1810	HR_CS_MQTT_BROKER_LONG_ PASSWORD_31		0x0000	0xFFFF	RW
1811	HR_CS_MQTT_BROKER_LONG_ PASSWORD_32		0x0000	0xFFFF	RW

Table 17

12.3 ACCESS TO CIRCULAR BUFFER

DigiRail IoT has a circular buffer to log events and periodic collections. In periodic collections, the data from all enabled channels will be saved. In Event mode, only the data from the event that originated the log will be saved. Through the configuration, it is possible to add the count value even when the channel is set as Event.
Three Modbus registers indicate the collection positions and allow you to read the circular buffer: HR_SS_COLLECT_RECORD_MAX_QTTY, HR_SS_COLLECT_LAST_RECORD, and HR_SS_COLLECT_FIRST_RECORD. The application must rely on these 3 registers to calculate the position to be requested (see function ***u16GetNextIndex*** from the example code below).

After calculating the position, the application must write the value to the Modbus register HR_SS_COLLECT_REQUESTED_RECORD. After writing the value, the application can read the registers, which have already been updated with the requested collection values. To monitor and download the registers from memory, see the ***TaskReadMem*** function from the example code below.

Address 1 is the first valid read position.

If you request a position with no data to collect yet, all registers with values intended for download will be filled with the value 0xFFFF.

12.3.1 CIRCULAR BUFFER: REGISTER TABLE

Below is the table of registers for circular buffer:

	ADDRESS	NAME	DESCRIPTION	PERMISSION
COLLECTION POSITION CALCULATION	187	HR_SS_COLLECT_RECORD_MAX_QTTY	Maximum number of collections supported by memory.	RO
	188	HR_SS_COLLECT_LAST_RECORD	Position of the last collection added to memory.	RO
	189	HR_SS_COLLECT_FIRST_RECORD	Position of the first collection added to memory.	RO
REQUEST COLLECTION	190	HR_SS_COLLECT_REQUESTED_RECORD	Position of the collection requested for the reading.	RW
REQUESTED REGISTER DATA	191	HR_SS_COLLECT_TIMESTAMP_UNIX_H	Timestamp of the requested download in Unix format.	RO
	192	HR_SS_COLLECT_TIMESTAMP_UNIX_L		RO
	193	HR_SS_COLLECT_TIMESTAMP_MS	Timestamp of the requested download in milliseconds.	RO
	194	HR_SS_COLLECT_CHD_EVENT_INDEX	Index of the digital channel when an event occurs.	RO
	195	HR_SS_COLLECT_CHD_EVENT_TYPE	Event type (when it occurs).	RO
	196	HR_SS_COLLECT_CHD1_VALUE_H	Digital channel 1 counter value.	RO
	197	HR_SS_COLLECT_CHD1_VALUE_L		RO
	198	HR_SS_COLLECT_CHD2_VALUE_H	Digital channel 2 counter value.	RO
	199	HR_SS_COLLECT_CHD2_VALUE_L		RO
	200	HR_SS_COLLECT_CHD3_VALUE_H	Digital channel 3 counter value.	RO
	201	HR_SS_COLLECT_CHD3_VALUE_L		RO
	202	HR_SS_COLLECT_CHD4_VALUE_H	Digital channel 4 counter value.	RO
	203	HR_SS_COLLECT_CHD4_VALUE_L		RO
	204	HR_SS_COLLECT_CHD5_VALUE_H	Digital channel 5 counter value.	RO
	205	HR_SS_COLLECT_CHD5_VALUE_L		RO
	206	HR_SS_COLLECT_CHD6_VALUE_H	Digital channel 6 counter value.	RO
	207	HR_SS_COLLECT_CHD6_VALUE_L		RO
	208	HR_SS_COLLECT_CH1_SENSE_USER_RANGE_H	Displays the sensor value in the user range of analog channel 1 (in Float).	RO
	209	HR_SS_COLLECT_CH1_SENSE_USER_RANGE_L		RO
	210	HR_SS_COLLECT_CH2_SENSE_USER_RANGE_H	Displays the sensor value in the user range of analog channel 2 (in Float).	RO
	211	HR_SS_COLLECT_CH2_SENSE_USER_RANGE_L		RO

Table 18

12.3.2 CIRCULAR BUFFER: AVAILABILITY TABLE

The table below is used to evaluate the impact on the maximum number of collections supported depending on the channels that are enabled and if digital channel counting in events is also enabled:

DIGITAL CHANNELS	ANALOG CHANNELS	MAXIMUM AMOUNT (WITHOUT EVENT COUNTING)	MAXIMUM AMOUNT (WITH EVENT COUNTING)
0	1	7096	4913
0	2	5806	4913
1	0	5806	4913
2	0	4258	4258
3	0	3361	3361
4	0	2777	2777
5	0	2365	2365
6	0	2060	2060
6	1	1935	1935
6	2	1824	1824

Table 19

12.3.3 EXAMPLE CODE

```
typedef enum e_collect_memmap
{
        ADDR_MAX_RECORDS_QTTY,
        ADDR_LAST_RECORD,
        ADDR_FIRST_RECORD,
        ADDR_REQUESTED_RECORD,
        ADDR_TIMESTAMP_UNIX_HIGH,
        ADDR_TIMESTAMP_UNIX_LOW,
        ADDR_TIMESTAMP_MS,
        ADDR_DIGITAL_CHANNEL_EVENT_INDEX,
        ADDR_EVENT_TYPE,
        ADDR_DIGITAL_CHANNEL_1_HIGH,
        ADDR_DIGITAL_CHANNEL_1_LOW,
        ADDR_DIGITAL_CHANNEL_2_HIGH,
        ADDR_DIGITAL_CHANNEL_2_LOW,
        ADDR_DIGITAL_CHANNEL_3_HIGH,
        ADDR_DIGITAL_CHANNEL_3_LOW,
        ADDR_DIGITAL_CHANNEL_4_HIGH,
        ADDR_DIGITAL_CHANNEL_4_LOW,
        ADDR_DIGITAL_CHANNEL_5_HIGH,
        ADDR_DIGITAL_CHANNEL_5_LOW,
        ADDR_DIGITAL_CHANNEL_6_HIGH,
        ADDR_DIGITAL_CHANNEL_6_LOW,
        ADDR_ANALOG_CHANNEL_1_HIGH,
        ADDR_ANALOG_CHANNEL_1_LOW,
        ADDR_ANALOG_CHANNEL_2_HIGH,
        ADDR_ANALOG_CHANNEL_2_LOW
} collect_memmap_t;

typedef enum e_digital_channels
{
        DIGITAL_CHANNEL_1,
        DIGITAL_CHANNEL_2,
        DIGITAL_CHANNEL_3,
        DIGITAL_CHANNEL_4,
        DIGITAL_CHANNEL_5,
        DIGITAL_CHANNEL_6,
        DIGITAL_CHANNELS_TOTAL
} digital_channels_t

typedef enum e_analog_channels
{
        ANALOG_CHANNEL_1,
        ANALOG_CHANNEL_2,
        ANALOG_CHANNELS_TOTAL
} analog_channels_t

typedef enum e_channel_digital_event_index
{
    DIGITAL_CHANNEL_EVENT_INDEX_NONE, // Periodic log, no event associated to digital channel
    DIGITAL_CHANNEL_EVENT_INDEX_CH1, // Event - channel 1
    DIGITAL_CHANNEL_EVENT_INDEX_CH2, // Event - channel 2
    DIGITAL_CHANNEL_EVENT_INDEX_CH3, // Event - channel 3
        DIGITAL_CHANNEL_EVENT_INDEX_CH4, // Event - channel 4
        DIGITAL_CHANNEL_EVENT_INDEX_CH5, // Event - channel 5
        DIGITAL_CHANNEL_EVENT_INDEX_CH6, // Event - channel 6
} channel_digital_event_index_t;
```

```
typedef enum e_event_type
{
        EVENT_TYPE_NONE,
        EVENT_TYPE_FALLING_EDGE,
        EVENT_TYPE_RISING_EDGE,
} event_type_t;

#define COLLECTED_DATA_SIZE 21

/*****************************************************************//**
 * @brief        Gets the next record index to be requested based on the last record already collected
 * @param[in]    actualIndex    Record index from register already collected
 * @return       uint16_t             Next record index to be collected
 *****************************************************************/
uint16_t u16GetNextIndex(uint16_t actualIndex)
{
        uint16_t lastRecord = FncReadSingleRegisterModbus(ADDR_LAST_RECORD);
        uint16_t firstRecord = FncReadSingleRegisterModbus(ADDR_FIRST_RECORD_REGISTER);
        uint16_t recordsQtty = FncReadSingleRegisterModbus(ADDR_MAX_RECORDS_QTTY);

        // when the index of collected record is different from the index of last record in memory
        if (actualIndex != lastRecord)
        {
                // no record has been overwritten
                if (lastRecord > firstRecord)
                {
                        // collected record index is less than the index of last record in memory
                        if (actualIndex < lastRecord)
                        {
                                return actualIndex + 1;
                        }
                }
                // records circulated the memory
                else if (lastRecord < firstRecord)
                {
                        // collected record index is less than the index of last record in memory
                        if (actualIndex < lastRecord)
                        {
                                return actualIndex + 1;
                        }
                        // collected record index is higher than the most recent record and LESS than
memory capacity
                        else if (actualIndex < recordsQtty)
                        {
                                return actualIndex + 1;
                        }
                        // collected record index is higher than the most recent record and HIGHER than
memory capacity
                        else
                        {
                                return 1; // first record address
                        }
                }
        }

        return actualIndex;
}

/*****************************************************************//**
 * @brief        Thread to monitor new records and collect when needed
 * @param[in]    None
 * @return       None
 *****************************************************************/
void TaskReadMem (void)
{
        uint16_t actualIndex = 0, nextIndex, lastRecord, buf[COLLECTED_DATA_SIZE];

        while (1)
        {
       // reads the index of the last record in memory
                lastRecord = FncReadSingleRegisterModbus(ADDR_LAST_RECORD);

                // if the index of collected record is different from the index of last record in memory
                if (lastRecord != actualIndex)
                {
                        nextIndex = u16GetNextIndex(actualIndex);

                        // requests a record by writing the index through a modbus register
                        FncWriteSingleRegisterModbus(ADDR_REQUESTED_RECORD, nextIndex);

                        // collects record data from requested index
                        FncReadBufferModbus(buf, ADDR_TIMESTAMP_UNIX_HIGH, COLLECTED_DATA_SIZE);

                        // after app uses the record, should update the index
                        if (FncUseColletedData(buf) == FNC_SUCCESS)
                        {
                                actualIndex = nextIndex;
                        }
```

```
                }
                threadSleep(100);
        }
}

/*
 * Functions that require user implementation
 *
 * FncReadSingleRegisterModbus (uint16_t registerAddr)
 * FncReadBufferModbus (uint16_t* buffer, uint16_t registerInitAddr, uint16_t size)
 * FncWriteSingleRegisterModbus  (uint16_t registerAddr, uint16_t value)
 * FncUseColletedData (uint16_t* collectedData)
 *
 */
```

12.3.4 CIRCULAR BUFFER: EXAMPLES

EXAMPLE 1

In the example below, not enough logs have yet been generated to circulate in the memory:

Position	Memory
1	Log 1
2	Log 2
3	Log 3
4	Log 4
5	Log 5
6	Log 6
7	Log 7
8	
9	
10	
	Position
First log	1
Last log	7

Figure 42

EXAMPLE 2

In the example below, the new logs have already circulated in the memory:

Position	Memory
1	Log 11
2	Log 12
3	Log 13
4	Log 4
5	Log 5
6	Log 6
7	Log 7
8	Log 8
9	Log 9
10	Log 10
	Position
First Log	4
Last Log	3

Figure 43

EXAMPLE 3

In the example below, the memory already circulated has advanced:

Position	Memory
1	Log 11
2	Log 12
3	Log 13
4	Log 14
5	Log 15
6	Log 6
7	Log 7
8	Log 8
9	Log 9
10	Log 10
	Position
First Log	6
Last Log	5

Figure 44

12.3.5 COLLECTION: EXAMPLES

EXAMPLE 4

Rising event on digital channel 2:

Register	Value
TIMESTAMP_UNIX_HIGH	0x607F
TIMESTAMP_UNIX_LOW	0x540A
TIMESTAMP_MS	300
DIGITAL_CHANNEL_EVENT_INDEX	2
EVENT_TYPE	1
DIGITAL_CHANNEL_1_ACC_HIGH	0xFFFF
DIGITAL_CHANNEL_1_ACC_LOW	0xFFFF
DIGITAL_CHANNEL_2_ACC_HIGH	0x001A
DIGITAL_CHANNEL_2_ACC_LOW	0x5648
DIGITAL_CHANNEL_3_ACC_HIGH	0xFFFF
DIGITAL_CHANNEL_3_ACC_LOW	0xFFFF
DIGITAL_CHANNEL_4_ACC_HIGH	0xFFFF
DIGITAL_CHANNEL_4_ACC_LOW	0xFFFF
DIGITAL_CHANNEL_5_ACC_HIGH	0xFFFF
DIGITAL_CHANNEL_5_ACC_LOW	0xFFFF
DIGITAL_CHANNEL_6_ACC_HIGH	0xFFFF
DIGITAL_CHANNEL_6_ACC_LOW	0xFFFF
ANALOG_CHANNEL_1_USER_RANGE_HIGH	0xFFFF
ANALOG_CHANNEL_1_USER_RANGE_LOW	0xFFFF
ANALOG_CHANNEL_2_USER_RANGE_HIGH	0xFFFF
ANALOG_CHANNEL_2_USER_RANGE_LOW	0xFFFF

Figure 45

EXAMPLE 5

Periodic log with digital channels 3 and 6 disabled, as well as analog channel 1:

Register	Value
TIMESTAMP_UNIX_HIGH	0x607F
TIMESTAMP_UNIX_LOW	0x5511
TIMESTAMP_MS	889
DIGITAL_CHANNEL_EVENT_INDEX	0
EVENT_TYPE	0
DIGITAL_CHANNEL_1_ACC_HIGH	0x0000
DIGITAL_CHANNEL_1_ACC_LOW	0x0001
DIGITAL_CHANNEL_2_ACC_HIGH	0x001A
DIGITAL_CHANNEL_2_ACC_LOW	0x5648
DIGITAL_CHANNEL_3_ACC_HIGH	0xFFFF
DIGITAL_CHANNEL_3_ACC_LOW	0xFFFF
DIGITAL_CHANNEL_4_ACC_HIGH	0x0000
DIGITAL_CHANNEL_4_ACC_LOW	0x0841
DIGITAL_CHANNEL_5_ACC_HIGH	0x7800
DIGITAL_CHANNEL_5_ACC_LOW	0x1566
DIGITAL_CHANNEL_6_ACC_HIGH	0xFFFF
DIGITAL_CHANNEL_6_ACC_LOW	0xFFFF
ANALOG_CHANNEL_1_USER_RANGE_HIGH	0xFFFF
ANALOG_CHANNEL_1_USER_RANGE_LOW	0xFFFF
ANALOG_CHANNEL_2_USER_RANGE_HIGH	0xC1BC
ANALOG_CHANNEL_2_USER_RANGE_LOW	0x0000

Figure 46

13 APPENDIX 3 – MQTT PROTOCOL

13.1 INTRODUCTION

This document describes the required infrastructure, the data published by the device, and the operation mode of the **DigiRail IoT**, which can publish data to the cloud via the MQTT protocol. The device provides support for the following set of MQTT Brokers:

- Google IoT
- Microsoft Azure
- AWS
- **NOVUS Cloud**
- Generic MQTT Broker (version 5.0 or higher).

13.2 PUBLISH AND SUBSCRIPE TOPICS

DigiRail IoT uses 5 topics. These topics are defined in the device configuration process and stored in the following variables:

- **Device data:** Used to publish the data generated in the device. It has 2 types: **`channel`** and **`events`**.
- **Config:** Used to send configuration data to the device. The device subscribes to this topic and shows the updates in the **Config Ack** topic.
- **Config Ack:** The device publishes the current configuration in this topic. If the **Config** topic receives a new configuration, this topic will confirm whether the new configuration has been applied.
- **Command:** The device receives (subscribe) commands through this topic. The result of this command is published in the **Command Ack** topic.
- **Command Ack:** The device publishes the result of commands executed in this topic.

See the table below:

TOPIC	PUB/SUB	USE
Device data	*Publish*	Publishes the data generated by the device. This topic receives Channel Data and Events.
Config	*Subscribe*	Receives the configuration data.
Config Ack	*Publish*	Responds to the configuration data.
Command	*Subscribe*	Receives commands. This topic receives Output, Reset counters, RS485 MQT Gateway and Get diagnostic commands.
Command Ack	*Publish*	Responds to the execution of the commands.

Table 20

13.2.1 PUBLISH BASIC MODEL

To simplify the treatment of MQTT message content, publications will always display the product model identifier and the user-defined identifier, labeled by the "**`pid`**" and "**`device_id`**" fields, respectively. The value of the "**`device_id`**" field is set in the Device ID parameter of the MQTT settings of the **NXperience** software.

Applicable **DigiRail IoT** Identifiers:

MODEL	PID
DigiRail IoT ETH	51584019
DigiRail IoT WRL	51518482

Table 21

13.2.2 DATE AND EVENT PUBLISHING MODEL

Publishing of the events and data generated by the device follows the standard MQTT template and uses a topic defined during configuration.

13.2.3 CONFIGURATION AND COMMAND PUBLISHG MODEL

The basic model of how the commands and settings work was based on the device twins implementation of the Microsoft Azure cloud, which, as described in Understand and use device twins in IoT Hub, is used to synchronize device settings and conditions.

This model has two basic concepts:

- **`Desired properties:`** These are the conditions and settings that the backend application can change or query on the device it is interacting with.
- **`Reported properties:`** These are used as a response after receiving Desired properties. The device reports the status or the result of a command.

This message exchange model needs two different topics to work. The first is the topic in which the device is subscribed to receive **`Desired properties`**. This step, initiated by the application, is called "**`request`**". In the second topic, the device will publish the **`Reported properties`** after executing a command or applying a configuration. This step is called "**`response`**".

13.3 DATA AND EVENTS

The data will be published to the topic defined in the variable **`Device data`**. The data type is indicated in the JSON of the message. You should note the following items for all data:

13.3.1 CHANNEL DATA

The channel data is published periodically, according to the device configuration. The data is in JSON format and has the following key/value sets:

```
{
    "pid": 51387408,
    "device_id": "device0",
    "channels" : {
        "timestamp":1585819219,
        "chd1_value":0,
        "chd2_value":0,
        "chd3_value":0,
        "chd4_value":0,
        "chd5_value":0,
        "chd6_value":0,
        "ch1_user_range":2.17,
        "ch2_user_range":2.2
    }
}
```

Notes:

- **`timestamp`** is in UTC.

13.3.2 EVENTS

The event data will be published whenever a previously configured event in the device occurs. The data is in JSON format and has the following key/value sets:

```
{
    "pid": 51387408,
    "device_id": "device0",
    "events": {
        "chd1": {
            "timestamp":1585819219.685,
            "edge":1,
        }
    }
}
```

Notes:

- The **`timestamp`** value is also in UTC, but in double format, with the milliseconds of the event in the fractional part.

13.4 CONFIGURATIONS

You can change or query the device settings by publishing to the topic defined in the **Config** variable. In the **Config Ack** topic, you can see if the changes have been executed and query their status.

The configuration items for this type of device are as follows:

CONFIGURATION ITEMS	DESCRIPTION
rtc	Setting of the RTC (Real Time Clock).
device	General device configuration.
chdX	Configuration of the digital channel 'X' (Available: **chd1**, **chd2**, **chd3**, **chd4**, **chd5**, and **chd6**).
periodic counter reset	Configuration of the reset periodicity of the digital counters.
chX	Configuration of the analog channel 'X' (Available: **ch1** and **ch2**).
chrX	Configuration of the remote channel 'X' (Available: **chr1**, **chr2**, **chr3**, **chr4**, **chr5**, **chr6**, **chr7**, and **chr8**).
eth	Network parameters configuration.
wifi	Wi-Fi interface configuration (if available).
ntp	NTP server configuration for automatic clock adjust.
modbus tcp	Modbus-TCP protocol configuration.
rs485	RS485 interface configuration.

Table 22

13.4.1 HOW TO CHANGE THE CONFIGURATION PARAMETERS

The steps to change the configuration are as follows:

STEP	ACTION
1	Send the configuration **request** when publishing to the **Config** topic.
2	The device, which is subscribed to the **Config** topic, evaluates and applies the new configuration.
3	The device publishes the response in the **Config Ack** topic.
4	The application, which is subscribed to the **Config Ack** topic, updates the device status and the result of the operation according to what it received in the response message.

Table 23

The data that was used when sending and receiving the configuration is in JSON format and is present in the payload of the messages that were exchanged between the application and the device.

The structure of the configuration **request** received by the device is as follows:

```
{
    "timestamp":1585819219,
    "desired":                                                    {
        <desired                                                item>
    }
}
```

The value given in **timestamp** is in UTC and serves to identify the configuration **request** message. The corresponding **response** will have the same value as the one received.

A **request** can contain only one item to be configured, called **<desired item>**, and you can send only the key/value pairs you want to change, omitting the others.

At the end of the execution, the device sends the response in the **Config Ack** topic, reporting the result of the operation for each configured item in the following format:

```
{
    "device_id":                                                "device0",
    "timestamp":1585819219,
    "reported":                                                   {
        <reported                                               item>
    }
}
```

The **timestamp** value in the **response** message is the same as in the configuration **request** message.

The configuration **request** publication can have only one **desired item** for each message. In most cases, the data structure of the **reported item** is the same as that of the **desired item**, but with the addition of an **item error** that indicates the result in the application of the respective **desired item**. Exceptions are indicated in each of the configuration items below.

13.4.2 HOW TO HANDLE ERRORS WHEN CHANGING THE CONFIGURATION

The values set in each **`desired item`** of the **`request`** will only be applied if the execution has no errors in any of the key/value pairs sent in that **`desired item`**. Each **`desired item`** is processed independently. There can be different response messages for each **`response item`**.

The **`error`** value is an integer and reports the first error encountered when applying the configuration for an item.

The table below shows the error codes:

CODE	DESCRIPTION
0	Success.
1	The structure is correct, but the device has received a parameter that is out of range.
2	The structure is correct, but the device received an unknown parameter.

Table 24

The table below shows the device actions for each error condition:

ERROR CONDITION	ACTION
Configuration item not recognized	The **`reported item`** will only contain the error value in **`error`**.
Error when applying a configuration	The **`reported item`** will contain the current configuration values and the error value will indicate the first error that occurred.

Table 25

13.4.3 HOW TO CONSULT THE CONFIGURATION PARAMETERS

The steps for querying the current configuration are as follows:

STEPS	ACTION
1	Send the **`request`**, indicating the configuration item that is to be queried in the **Config** topic.
2	The device, which is subscribed to the **Config** topic, evaluates the request, and reads the configuration data.
3	The device publishes the response in the **Config Ack** topic.
4	The application, which is subscribed to the **Config Ack** topic, updates the device status according to the data in the response message.

Table 26

The data that was used when sending and receiving the configuration is in JSON format and is present in the payload of the messages that were exchanged between the application and the device.

The structure of the configuration **request** received by the device is as follows:

```
{
    "timestamp":1585819219,
    "desired": {
        <empty desired item>
    }
}
```

The **`timestamp`** follows the default used when changing the configuration. As shown the above example, a **`request`** can contain only one item to be queried, called an <**`empty desired item`**>.

An **`empty desired item`** is a configuration item with no key/value pairs, as shown in the example below:

```
{
    "timestamp":1585819219,
    "desired": {
        "rtc" : {}
    }
}
```

In the example above, the corresponding **`response`** will have the value of the current RTC.

The configuration **`request`** publication can have multiple **`empty desired items`**, one for each item you want to query. The data structure of the **`reported items`** is the same as that used in the **`response`** you receive when changing parameters. If the queried item exists, the **`error`** value will indicate that the operation was successful.

13.4.4 HOW TO HANDLE ERRORS WHEN QUERYING THE CONFIGURATION

Each **empty desired item** is processed independently. The configuration **response** messages can have different return status. The **error** value is an integer and reports the first error encountered when reading the configuration of an item.

The table below shows the device actions for each error condition:

ERROR CONDITION	ACTION
Configuration item not recognized	The **reported item** will only contain the error value in **error**.
Error when applying a configuration	The **reported item** will contain the current configuration values and the error value will indicate the first error that occurred.

Table 27

13.4.5 CONFIGURATION ITEMS

RTC

REQUEST *RTC*	RESPONSE *RTC*

{
"timestamp":1585819219,
"desired": {
"rtc": {
"year":2021,
"month":2,
"day":25,
"hour":12,
"minute":13,
"sec":10
}
}
}
	{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"rtc": {
	"error": 0,
	"year":2021,
	"month":2,
	"day":25,
	"hour":12,
	"minute":13,
	"sec":10
	}
	}
	}

Table 28

DEVICE

REQUEST *DEVICE*	RESPONSE *DEVICE*

{
"timestamp":1585819219,
"desired": {
"device": {
"title":"Pci",
"location":"location_123",
"pub_interval":60,
"alter_pub_interval_enable":1,
"alter_pub_interval ~~enable~~":600,
"add_counter_on_events":1
}
}
}
	{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"device": {
	"error": 0,
	"title":"Pci",
	"location":"location_123",
	"pub_interval":60,
	"alter_pub_interval_enable":1,
	"alter_pub_interval ":600,
	"add_counter_on_events":1
	}
	}
	}

Table 29

DIGITAL CHANNELS

The example in this section shows only digital channel 1, indicated as **chd1**. The other channels (**chd2**, **chd3**, **chd4**, **chd5**, and **chd6**) follow the same data model.

REQUEST *DIGITAL CHANNELS*	RESPONSE *DIGITAL CHANNELS*

<pre>{
"timestamp":1585819219,
"desired": {
"chd1": {
"enable":1,
"counting_m":2,
"type":3,
"edge":1,
"debounce":555,
"reset_m":2,
"debounce_enable":0
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"chd1": {
	"error": 0,
	"enable":1,
	"counting_m":2,
	"type":3,
	"edge":1,
	"debounce":555,
	"reset_m":2
	"debounce_enable":0
	}
	}
	}</pre>

Table 30

PERIODIC COUNTER RESET

REQUEST *CHD_PERIODIC_RESET*	RESPONSE *CHD_PERIODIC_RESET*

<pre>{
"timestamp":1585819219,
"desired": {
"chd_periodic_reset" : {
"type":0,
"day":2,
"hour":3,
"minute":4,
"sec":5,
"week_day":6
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"chd_periodic_reset" : {
	"error": 0,
	"type":0,
	"day":2,
	"hour":3,
	"minute":4,
	"sec":5,
	"week_day":6
	}
	}
	}</pre>

Table 31

ANALOG CHANNELS

The example in this section shows only analog channel 1, indicated as **ch1**. The other channel (**chd2**) follows the same data model.

REQUEST *ANALOG CHANNELS*	RESPONSE *ANALOG CHANNELS*

<pre>{
"timestamp":1585819219,
"desired": {
"ch1" : {
"enable":1,
"sensor_type":1,
"range_min":-10,
"range_max":2020,
"decimal_point":2
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"ch1" : {
	"error": 0,
	"enable":1,
	"sensor_type":1,
	"range_min":-10,
	"range_max":2020,
	"decimal_point":2
	}
	}
	}</pre>

Table 32

REMOTE CHANNELS

The example in this section shows only remote channel 1, indicated as **chr1**. The other channels follow the same data model.

REQUEST *REMOTE CHANNELS*	RESPONSE *REMOTE CHANNELS*

<pre>{
"timestamp":1585819219,
"desired": {
"chr1" : {
"enable":1,
"addr_slave":1,
"addr_register":123,
"cmd_modbus":1,
"data_type":0,
"decimal_point":0,
"bytes_inversion":0,
"error_value":9999,
"error_enable":0
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"chr1" : {
	"error": 0,
	"addr_slave":1,
	"addr_register":123,
	"cmd_modbus":1,
	"data_type":0,
	"decimal_point":0,
	"bytes_inversion":0,
	"error_value":9999,
	"error_enable":0
	}
	}
	}</pre>

Table 33

Note:

- To change the value of the **data_type** or **error_value** keys, you must send both keys in the same **request**.

ETHERNET

REQUEST *ETHERNET*	RESPONSE *ETHERNET*

<pre>{
"timestamp":1585819219,
"desired": {
"eth" : {
"enable_dhcp":0,
"addr":[10, 167, 2, 3],
"mask":[255,255, 255, 0],
"gateway":[255, 255, 255, 0],
"ipv4dns":[8, 8, 8, 8]
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"eth" : {
	"error": 0,
	"enable_dhcp":0,
	"addr":[10, 167, 2, 3],
	"mask":[255,255, 255, 0],
	"gateway":[255, 255, 255, 0],
	"ipv4dns":[8, 8, 8, 8]
	}
	}
	}</pre>

Table 34

WI-FI

REQUEST *ETHERNET*	RESPONSE *ETHERNET*

<pre>{
"timestamp":1585819219,
"desired": {
"wifi" : {
"ssid":"WifiName",
"pwd":"password"
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"wifi" : {
	"error": 0,
	"ssid":"WifiName"
	}
	}
	}</pre>

Table 35

Note:

- The **pwd** key is not transmitted in the response.

NTP

REQUEST *NTP*	RESPONSE *NTP*

<pre>{
"timestamp":1585819219,
"desired":{
"ntp":{
"enable":1,
"diff_to_update":5,
"host":"time.google.com"
}
}
}</pre>
	<pre>{
	"pid":51387408,
	"device_id":"digirail_iot",
	"timestamp":1585819219,
	"reported":{
	"ntp":{
	"error":0,
	"enable":1,
	"diff_to_update":5,
	"host":"time.google.com"
	}
	}
	}</pre>

Table 36

MODBUS-TCP

REQUEST *MODBUS-TCP*	RESPONSE *MODBUS-TCP*

<pre>{
"timestamp":1585819219,
"desired": {
"modbus_tcp" : {
"enable":1,
"port":502
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	" reported ": {
	"modbus_tcp" : {
	"error": 0,
	"enable":1,
	"port":502
	}
	}
	}</pre>

Table 37

RS485

REQUEST *RS485*	RESPONSE *RS485*

<pre>{
"timestamp":1585819219,
"desired": {
"rs485" : {
"baudrate":6,
"stopbits":1,
"parity":1,
"timeout":500,
"oper_mode":1,
"addr_rtu":1,
"num_read":1,
"int_read":100
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"rs485" : {
	"error": 0,
	"baudrate":6,
	"stopbits":1,
	"parity":1,
	"timeout":500,
	"oper_mode":1,
	"addr_rtu":1,
	"num_read":1,
	"int_read":100
	}
	}
	}</pre>

Table 38

Note:

- The **`timeout`** and **`int_read`** keys have a value in milliseconds.

13.5 COMMANDS

The data will be published to the topic defined in the variable **Command**. The data type is indicated in the JSON of the message. The return from the execution of the commands occurs through the **Command Ack** topic.

13.5.1 OUTPUT

This command changes the status of the device outputs.

REQUEST *OUTPUT*	RESPONSE *OUTPUT*

<pre>{
"timestamp":1585819219,
"desired": {
"output" : {
"out1":1,
"out2":1
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"output" : {
	"error": 0,
	"out1":1,
	"out2":1
	}
	}
	}</pre>

Table 39

Note:

- Status that will not be modified do not need to be published.

Notes:

- The **`timestamp`** is the same as the received command.
- The status described in **`desired`** will only be applied if execution is done without errors.
- The value of **`error`** is an integer and reports the first error found during the execution of the command.
- If the command failed, the status reported will be the current.

13.5.2 RESET COUNTERS

This command resets the digital channel counters.

REQUEST *RESET COUNTERS*	RESPONSE *RESET COUNTERS*

<pre>{
"timestamp":1585819219,
"desired": {
"reset_counters" : {
"reset_chd2":1,
"reset_chd4":1,
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported" : {
	"reset_counters": {
	"error": 0,
	"reset_chd1":0,
	"reset_chd2":0,
	"reset_chd3":0,
	"reset_chd4":0,
	"reset_chd5":0,
	"reset_chd6":0
	}
	}
	}</pre>

Table 40

Note:

- Counters that will not be reset do not need to be published

Notes:

- The **`timestamp`** is the same as the received command.
- The status described in **`desired`** will only be applied if execution is done without errors.
- The value of **`error`** is an integer and reports the first error found during the execution of the command.
- The **`reset_chdX`** keys (with X from 1 to 6) can assume values 0 or 1. When the value is 1, the counter will be reset. The value 0 indicates that the counter should not be changed.

13.5.3 SET COUNTERS

This command changes the value of the digital channels' counters.

REQUEST *SET COUNTERS*	RESPONSE *SET COUNTERS*

<pre>{
"timestamp":1620413979
"desired": {
"set_counters" : {
"set_chd2":6500,
"set_chd3":10
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1620413979,
	"reported" : {
	"set_counters": {
	"error": 0,
	"set_chd1":0,
	"set_chd2":6500,
	"set_chd3":10,
	"set_chd4":0,
	"set_chd5":0,
	"set_chd6":0
	}
	}
	}</pre>

Table 41

Nota:

- Counters that will not be changed should not be published.

Notes:

- The **`timestamp`** is the same as the command received (**`desired`**).
- The status described in the **`desired`** step will only be applied if the execution is done without errors.
- The **`error`** value is an integer and reports the error found during the command execution.
- In this example, digital channels 1, 4, 5 and 6 do not appear in the JSON **`desired`** since you do not want to change their counters. In the response, the current value of the digital channel will be returned. For digital channels 1, 4, 5, and 6 the current value is assumed to be zero.

13.5.4 GATEWAY MQTT RS485

This command sends the bytes of mb_buffer over the RS485 interface. The value of each byte contained in mb_buffer must be in hexadecimal format.

REQUEST *GATEWAY MQTT RS485*	RESPONSE *GATEWAY MQTT RS485*

<pre>{
"timestamp":15,
"desired": {
"gateway_485": {
"mb_buffer":"02 03 00 00 00 0A C5 FE"
}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id":"DeviceName",
	"timestamp":15,
	"reported": {
	"gateway_485": {
	"error":0,
	"mb_buffer":"02 03 14 19 C7 00 00 06 4E 00 00 04 E0 00 00 03 D0 00 00 03 D0 00 00 1B 13"
	}
	}
	}</pre>

Table 42

Notes:

- In the **`response`** step of the MQTT command, the bytes received on the RS485 interface are contained in mb_buffer.
- If the device timeout is addressed to the RS485 interface, mb_buffer will return empty.

Notes:

- The **`timestamp`** is the same as the received command.
- The value of **`error`** is an integer and reports the error found during the execution of the command

13.5.5 GET DIAGNOSTIC

REQUEST *GET DIAGNOSTIC*	RESPONSE *GET DIAGNOSTIC*

<pre>{
"timestamp":1585819219,
"desired" : {
"diag" : {}
}
}</pre>
	<pre>{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported" : {
	"diag": {
	"title": "Pci v2",
	"location":"home",
	"curr_timestamp":1589326517,
	"cfg_timestamp":1589311676,
	"fw_v":"01.00",
	"mqtt_queue":1,
	"sn":"00000001",
	"curr_rssi":"55",
	"min_rssi":"46",
	"max_rssi":"87",
	"avg_rssi":"54",
	"ipv4":[192, 168, 0, 23]
	}
	}
	}</pre>

Table 43

If the **Publish Diagnostics Periodically** parameter of the **NXperience** configuration software (see the MQTT PROTOCOL section of the CONFIGURATION SOFTWARE chapter) is enabled, the occurrence counters for the system events will also be added to the response:


```
{
   "pid":51387408,
   "device_id":"digirail_iot",
   "timestamp":1585819219,
   "reported":{
      "diag":{
         "error":0,
         "title":"Pci v2",
         "location":" home ",
         "curr_timestamp":1589326517,
         "cfg_timestamp":1589311676,
         "fw_v":"1.23",
         "mqtt_queue":1,
         "sn":"00000001",
         "curr_rssi":"55",
         "min_rssi":"45",
         "max_rssi":"70",
         "avg_rssi":"55",
         "ipv4":[
            192,
            168,
            0,
            23
         ],
         "log_counters":{
            "pwr_on":1,
            "pwr_sw_reset":0,
            "net_disconnected":1,
            "wifi_prov_error":0,
            "dhcp_error":0,
            "dns_error_1":0,
            "dns_error_2":0,
            "cfg_updated":1,
            "fw_updated":0
         },
         "watchdog_counters":{
            "analog":"0",
```

```
            "data_storage":"0",
            "record_storage":"0",
            "digital":"0",
            "modbus":"0",
            "record_periodic":"0",
            "mqtt":"1",
            "network":"0"
        }
    }
  }
}
```


13.5.6 RESET DIAGNOSTIC

The **reset diag** command is used so that the application can reset counters related to internal system events and Wi-Fi signal quality (RSSI) measurement data.

The values of the **reset_watchdog_counter**, **reset_x_counter** and **reset_diag_rssi** fields can have values of 0 or 1. The value "1" means that a reset is to be applied to the corresponding parameter. The value "0" indicates that the parameter should not be changed. In this case you can also simply omit the JSON channel.

REQUEST *RESET DIAGNOSTIC*	RESPONSE *RESET DIAGNOSTIC*

{
"timestamp":1585819219,
"desired": {
"reset_diag": {
"reset_watchdog_counter":0,
"reset_logs_counter":1,
"reset_diag_rssi":1
}
}
}
	{
	"pid": 51387408,
	"device_id": "device0",
	"timestamp":1585819219,
	"reported": {
	"reset_diag": {
	"error": 0,
	"reset_watchdog_counter":0,
	"reset_logs_counter":0,
	"reset_diag_rssi":0
	}
	}
	}

Table 44

Notes:

- The **timestamp** is the same as the received command (**desired**).
- The status described in **desired** will only be applied if execution is done without errors.
- The value of **error** is an integer and reports the error encountered during the execution of the command.

13.5.7 LOGS

The logs command returns the last 50 log events from the system. All events will have an ID, which can be queried with this command, and a timestamp of the occurrence.

REQUEST *LOGS*	RESPONSE *LOGS*

{
"timestamp":1585819219,
"desired": {
"logs": {}
}
}
	{
	"pid":51387408,
	"device_id":"digirail_iot",
	"timestamp":1585819219,
	"reported":{
	"logs":{
	"error":0,
	"events":[
	{
	"ts":1638193059,
	"id":9
	},
	{
	"ts":1638193055,
	"id":10
	},

REQUEST *LOGS*	RESPONSE *LOGS*

	{
	"ts":1638192333,
	"id":9
	},
	{
	"ts":1636466491,
	"id":4
	}
	]
	}
	}
	}

Table 45

13.5.8 LOGS_PARSED

Due to device memory limitations, the **`logs_parsed`** command returns only the last 30 system log events. However, instead of giving an ID, there will be a short description of the log, plus the timestamp of the occurrence, like the `logs` command.

REQUEST *LOGS_PARSED*	RESPONSE *LOGS_PARSED*

{
"timestamp":1585819219,
"desired": {
"logs_parsed": {}
}
}
	{
	"device_id":"digirail_iot",
	"timestamp":1585819219,
	"reported":{
	"logs_parsed":{
	"error":0,
	"events":[
	{
	"ts":1638193059,
	"mqtt":"connected"
	},
	{
	"ts":1638193055,
	"mqtt":"disconnected"
	},
	{
	"ts":1638192333,
	"mqtt":"connected"
	},
	{
	"ts":1636468024,
	"net":"connected"
	}
	]
	}
	}
	}

Table 46

The table below shows a detailed description of the logs:

CODES	LOGS_PARSED		DESCRIPTION
0	pwr	on	Standard startup.
1	pwr	sw_reset	Startup triggered by software reset.
2	pwr	wdt_reset	Startup triggered by internal Watchdog.
3	pwr	lvd_reset	Startup triggered by power outage.
4	net	connected	Connected to a network (Wi-Fi or Ethernet).
5	net	disconnected	Disconnected from the network (Wi-Fi or Ethernet).
6	wifi	prov_error	Wi-Fi provisioning failure (SSID or password incorrect).
7	dhcp	error	DHCP error.
8	sntp	error	SNTP error.
9	mqtt	connected	Connected to a MQTT broker.
10	mqtt	disconnected	Disconnected from the MQTT broker.
11	mqtt	sub_error	MQTT topics subscription error.
12	mqtt	pub_error	MQTT topics publishing error.
13	mqtt	alter_int	Publish interval has been changed to an alternative interval.
14	mqtt	default_int	Publish interval has been changed to a default interval.
15	dns	error_1	DNS internal error - Phase 1.
16	dns	error_2	DNS internal error - Phase 2.
17	dns	error_3	DNS internal error - Phase 3.
18	mem	init_error	There was an error during the circular buffer initialization. Device has recovered.
19	mem	not_init	The circular buffer has not been initialized.
20	mem	read_error	There was a failure while reading the circular buffer.
21	cfg	updated	Device configuration updated.
22	fw	updated	Device firmware updated.

Table 47

13.6 TOPICS IN MULTIPLE CLOUDS

The topics used by the device will be configured according to the cloud type you select. Topics are unique to one device, which is identified by the **`{id}`** variable. This variable is supplied during the configuration process.

13.6.1 AWS

VARIABLE	TOPIC
Device data	NOVUS/{id}/events
Config	NOVUS/{id}/config
Config Ack	NOVUS/{id}/ack/config
Command	NOVUS/{id}/command
Command Ack	NOVUS/{id}/ack/command

Table 48

13.6.2 GOOGLE IOT

VARIABLE	TOPIC
Device data	/devices/{id}/events
Config	/devices/{id}/commands/#
Config Ack	/devices/{id}/events
Command	/devices/{id}/commands/#
Command Ack	/devices/{id}/events

Table 49

13.6.3 MICROSOFT AZURE

VARIABLE	TOPIC
Device data	devices/{id}/events/
Config	devices/{id}/messages/devicebound/#
Config Ack	devices/{id}/events/
Command	devices/{id}/messages/devicebound/#
Command Ack	devices/{id}/events/

Table 50

13.6.4 NOVUS CLOUD

VARIABLE	TOPIC
Device data	NOVUS/{id}/events
Config	NOVUS/{id}/config
Config Ack	NOVUS/{id}/ack/config
Command	NOVUS/{id}/command
Command Ack	NOVUS/{id}/ack/command

Table 51

13.6.5 GENERIC BROKER

The parameters for a Broker that has been defined by the user can be set arbitrarily. To improve device performance, it is recommended to use the **Device data**, **Config Ack**, and **Command Ack** topics for publishing the device, and the **Config** and **Command** topics for publishing the device management/configuration application.

VARIABLE	TOPIC
Device data	Device Publish topic
Config	Device Subscribe topic
Config Ack	Device Publish topic
Command	Device Subscribe topic
Command Ack	Device Publish topic

Table 52

13.7 CONFIGURATION VARIABLES

These are the configuration variables allowed by the MQTT protocol:

CORRESPONDING CONFIGURATION ITEM	VARIABLE	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE
RTC	year	Allows you to configure the year to be used to set the RTC of the device.	2016	2080
	month	Allows you to configure the month to be used to set the RTC of the device.	1	12
	day	Allows you to configure the day to be used to set the RTC of the device.	1	28
	hour	Allows you to configure the hour to be used to set the RTC of the device.	0	23
	minute	Allows you to configure the minute to be used to set the RTC of the device.	0	59
	sec	Allows you to configure the second to be used to set the RTC of the device.	0	59
DEVICE	title	Allows you to define a name for the device.	-	20
	location	Allows you to inform where the device has been positioned.	-	40
	pub_interval	Allows you to configure the interval at which the data will be published to the MQTT Broker (in seconds).	1	65535
	alter_pub_interval_enable	Allows you to enable an alternative publishing interval whenever there are connection problems with the Broker MQTT.	0	1
	alter_pub_interval	Allows you to configure an alternative publishing interval whenever there are connection problems with the Broker MQTT.	60	65535
DIGITAL CHANNELS	enable	Allows you to enable the digital channel.	0	1
	counting_m	Allows you to configure the counting mode for the digital channel: 0 → Not defined 1 → Counter 2 → Event	0	2
	type	Allows you to configure the sensor type of the digital channel: 0 → Not configured 1 → PNP 2 → NPN 3 → Dry contact	0	3
	edge	Allows you to configure the counting edge of the digital channel: 1 → Rising edge 2 → Falling edge 3 → Both edges	1	3
	debounce	Allows you to configure the Debounce time of the digital channel for the Dry Contact sensor (in milliseconds).	0	60000
	reset_m	Allows you to configure the reset mode of the digital channel accumulators: Bit 0 → Overflow Bit 1 → Calendar Bit 2 → Protocol	0	2
	debounce_enable	Allows you to enable Debounce for the digital channel.	0	1
PERIODIC COUNTERS RESET	type	Allows you to configure the reset mode of the digital counters: 0 → Daily 1 → Weekly 2 → Monthly	0	2

CORRESPONDING CONFIGURATION ITEM	VARIABLE	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE
	day	Allows you to configure the reset day for the counter.	0	28
	hour	Allows you to configure the reset hour for the counter.	0	23
	minute	Allows you to configure the reset minute for the counter.	0	59
	sec	Allows you to configure the reset second for the counter.	0	59
	week_day	Allows you to configure the reset week for the counter.	1	7
ANALOG CHANNELS	enable	Allows you to enable the analog channel.	0	1
	sensor_type	Allows you to configure the sensor type of the analog channel: 0 → Not defined 0 → 0-5 V 2 → 0-10 V 3 → 0-20 mA 4 → 4-20 mA	0	4
	range_min	Allows you to configure the minimum limit of the analog channel.	0	0xFFFF
	range_max	Allows you to configure the maximum limit of the analog channel.	0	0xFFFF
	decimal_point	Allows you to configure the decimal place of the analog channel (fixed point for display and memory register): 0 → No decimal places 1 → One decimal place 2 → Two decimal places	0	2
ETHERNET	enable_dhcp	Allows you to define that the device gets its IP via DHCP.	0	1
	addr	Allows you to configure the IPv4 address of the device.	0	65535
	mask	Allows you to configure the network mask.	0	65535
	gateway	Allows you to configure the network Gateway.	0	65535
	ipv4dns	Allows you to configure the DNS server IP.	0	65535
WI-FI	ssid	Allows you to configure the Wi-Fi network SSID.	0x0000	0xFFFF
	password	Allows you to configure a Wi-Fi network password.	0x0000	0xFFFF
MODBUS-TCP	enable	Allows you to enable the Modbus-TCP protocol.	0	1
	port	Allows you to configure a communication port for the Modbus-TCP protocol.	0	0xFFFF
RS485	baudrate	Allows you to configure the Baud Rate of the RS485 interface: 0 → 1200 1 → 2400 2 → 4800 3 → 9600 4 → 19200 5 → 38400 6 → 57600 7 → 115200	0	7
	stopbits	Allows you to configure the Stop Bits of the RS485 interface: 0 → 1 Stop Bit 1 → 2 Stop Bits	0	1
	parity	Allows you to configure the parity of the RS485 interface: 0 → No parity	0	2

CORRESPONDING CONFIGURATION ITEM	VARIABLE	DESCRIPTION	MINIMUM VALUE	MAXIMUM VALUE
		1 → Odd parity 2 → Even parity		
	timeout	Allows you to configure a timeout value for the connection (in milliseconds).	0	65535
	oper_mode	Operating mode of the RS485 interface: 0 → Disabled 1 → Gateway mode 2 → Master mode 3 → Slave mode	0	3
	addr_rtu	Allows you to configure the address of the device on the Modbus-RTU network (when operating in slave mode).	1	247
	num_read	Number of attempts to read the slave devices (when operating in master mode).	1	65535
	int_read	Allows you to configure a time interval (in milliseconds) between sending one command and another (when operating in master mode).	0	65535
REMOTE CHANNELS	enable	Allows you to enable the remote channel.	0	1
	addr_slave	Allows you to configure the Modbus-RTU address of the slave device associated with the channel.	1	247
	addr_register	Allows you to define the starting address of the register to be read from the slave device.	0	65535
	cmd_modbus	Allows you to configure the Modbus read command: 0 → Rising edge 1 → Falling edge 2 → Both edges	0	3
	debounce	Allows you to configure the Debounce time of the digital channel configured for the Dry Contact sensor type (in milliseconds).	0	60000
	reset_m	Allows you to set the reset mode for the accumulators of digital channel 1: Bit 0 → Overflow Bit 1 → Calendar Bit 2 → Protocol.	0	2
	debounce_enable	Allows you to enable Debounce for the configured channel.	0	1

Table 53