DigiRail NXprog

INSTRUCTION MANUAL V1.0x G

novus
We Measure, We Control, We Record

CE UKCA

길우트레이딩 Since1984
서울특별시 영등포구 선유로 33길 15, 703호(양평동3가, 에이스테크노타워)
giltron@naver.com www.gilwoo.co.kr 02-2636-0009(대) FAX: 02-2636-4753

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 NXprog is an Arduino IDE programmable device that has analog and digital inputs and outputs and RS485 and Ethernet communication interfaces that meets the requirements for use in industrial environments the. With Arduino IDE, **DigiRail NXprog** allows the use of high-level programing languages such as C/C++ that enable complex algorithm implementation such as recursive logic, status machines, statistical analysis and mathematical equations and give greater flexibility for applications development.

Ideal for harsh environments, **DigiRail NXprog** combines the easy programming provided by Arduino IDE with the robustness required for devices for industrial use. For this purpose, **NOVUS** has incorporated features such as Watchdog Timer (WDT) and Brown-Out Detection (BOD), which are essential for any program on factory floor.

In addition to exclusive features for the Arduino software, all digital, analog and communication interfaces have protection devices to meet the strictest industry certification standards.

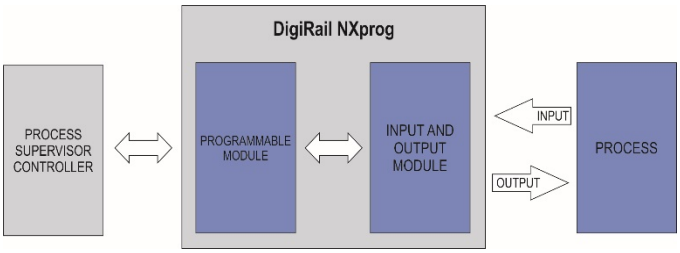

Figure 1 – DigiRail NXprog Process

DigiRail NXprog is internally composed of an input and output module and an Arduino IDE programmable module, which allows the user to develop their own application, and has four models with different input and output types: digital input channels (D), analog input channels (A), digital output channels (K), relay output channels (R) and analog output channels (O). The programmable module also has a real time clock (RTC) and a data memory (EEPROM) that allows developing a small data logger.

The configuration of the input and output module of **DigiRail NXprog** can be performed through an **NXperience** software configurator or an application developed in the Arduino IDE of the programmable module. Both allow defining the functions and mode of operation of the input and output channels and communication ports. In addition, **NXperience** enables values to be forced into analog and digital inputs and outputs and diagnostic analysis to be performed on the Ethernet interface and on the device.

It is recommended to use this manual to obtain information on the functionalities and configuration of the input and output module and to use the GitHub online documentation (https://github.com/NOVUS-Products/DigiRail-NXprog/) to obtain information on the specific functionalities of the programmable module. GitHub offers detailed information of how the programmable module works and shows examples of programs that can be loaded into the Arduino IDE.

3. IDENTIFICATION

3.1 DEVICE IDENTIFICATION

The identification of the device model is described on its side label, together with information regarding its power supply and its serial number. The figure below shows the information available in the device housing:

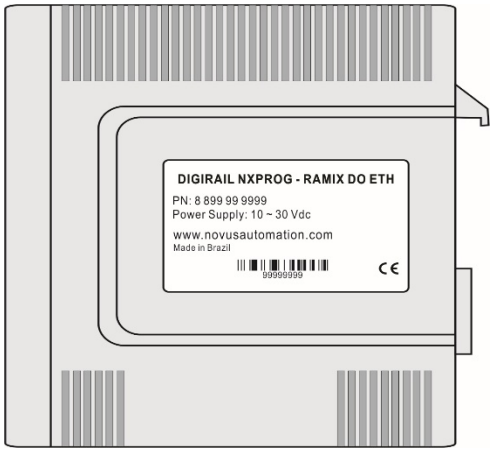

Figure 2 – DigiRail NXprog

3.2 DEVICE MODEL

DigiRail NXprog has 2 models: RAMIX DO ETH and RAMIX RL ETH. Both models have 1 USB port, 1 RS485 serial communication port and 1 Ethernet communication port.

The features of each model can be seen in the table below:

		Analog Input	Analog Output	Digital Input	Digital Transistor Output	Digital Relay Output
RAMIX	DO ETH	2	2	4	3	×
	RL ETH	2	2	4	×	2

Table 1 – DigiRail NXprog models

Figures 03 and **04** show the front of RAMIX DO ETH and RAMIX RL ETH models:

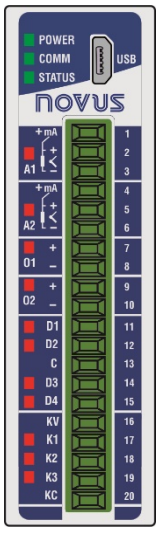

Figure 3 – DO ETH Model

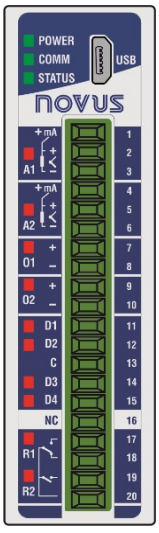

Figure 4 – RL ETH Model

4. INSTALLATION

4.1 MECHANICAL INSTALLATION

DigiRail NXprog is designed to have its housing fixed to a 35 mm DIN rail, as shown in the figure below. The 35 mm DIN rail installation must be carried out after the device has been configured.

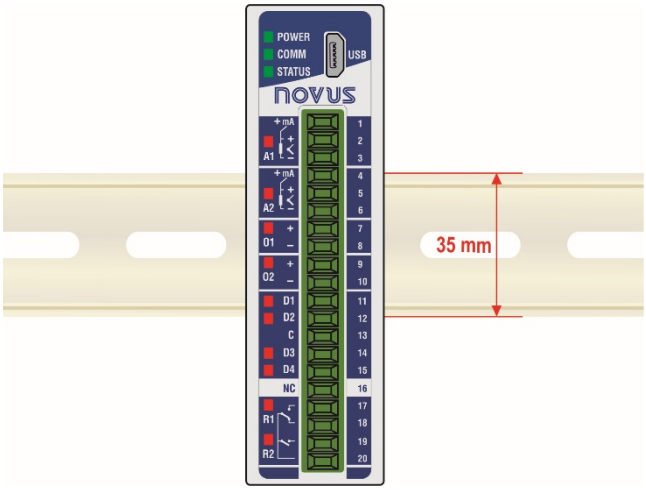

Figure 5 – Mechanical installation

4.2 DIMENSION

DigiRail NXprog has high quality housing, built in ABS+PC and with index of protection of IP20, which has the following dimensions:

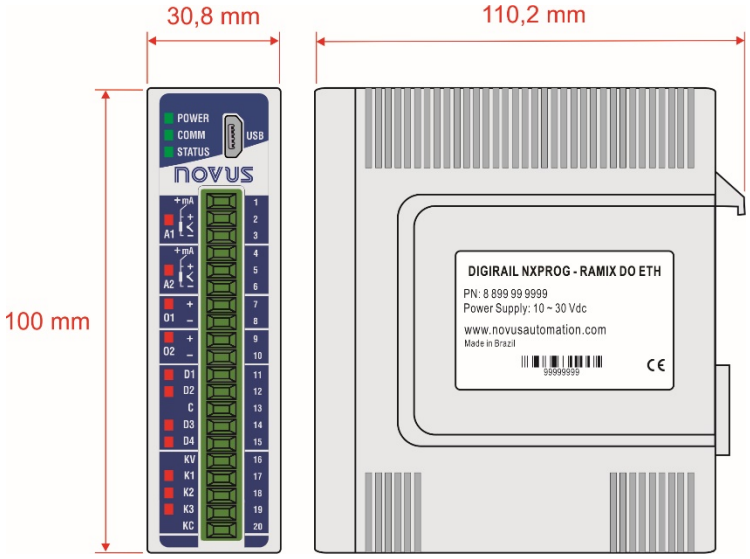

Figure 6 – Dimension

4.3 INSTALLATION RECOMMENDATIONS

- Electronic and analog signal drivers must run the plant separately from the output and power leads. If possible, in grounded conduits.
- The power supply for the electronic instruments must come from a proper power 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's internal security features do not guarantee full protection.
- The electrical connections must be made with the connection terminals marked on the device. Before connecting them, make sure that the connections have been made correctly.

5. CHARACTERISTICS AND CONNECTIONS

5.1 POWER SUPPLY CONNECTIONS AND COMMUNICATION PORTS

The power supply connections and communication ports can be viewed in the figure below:

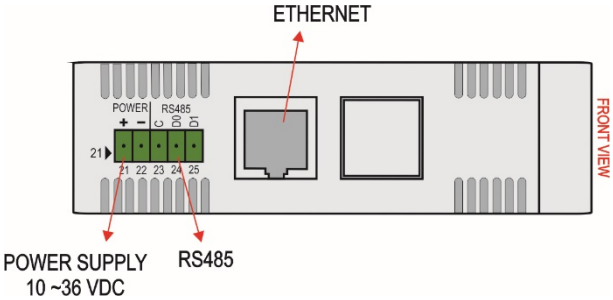

Figure 7 – Power supply and communication ports

DigiRail NXprog power terminals are located at the bottom of the housing and the polarization of this connection must be observed: Terminal 21 (+) and Terminal 22 (-).

5.1.1 USB CONNECTION

On its front panel, **DigiRail NXprog** provides a USB port, ideally intended for configuring and diagnosing the monitored process. During installation of the **NXperience** software, the USB port drivers will be automatically installed. During its first use, you must wait until Windows recognizes **DigiRail NXprog** driver.

	The USB port is NOT ISOLATED from the digital input and output circuits and the RS485 port circuit. **The USB interface allows only the configuration of the device. The RS485 interface and the analog and digital inputs and outputs will only work when the power supply is connected.**

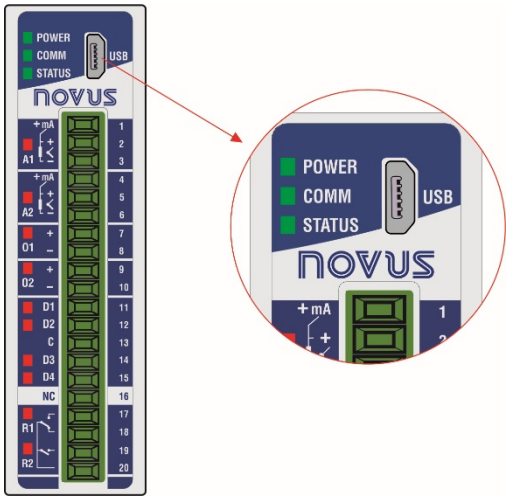

Figure 8 – USB connection

5.1.2 RS485 CONNECTION

The RS485 connection interface is located on the back of **DigiRail NXprog**, as shown in the figure below:

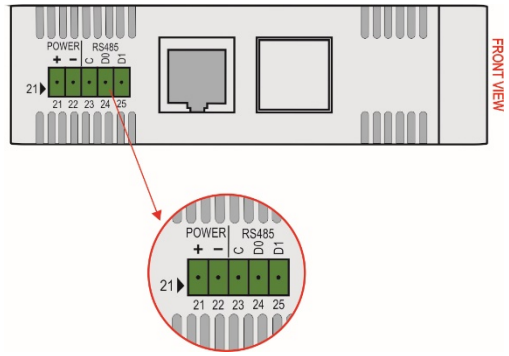

Figure 9 – RS485 connection

The RS485 interface can be configured to operate at the following Baud Rates: 1200, 2400, 4800, 9600, 19200, 38400, 57600 and 115200. It can also be configured to operate with 1 or 2 Stop Bits and parity even, odd and none. These parameters can be configured through **NXperience** software or through an Arduino application.

More details about the implementation of a Modbus devices network via RS485 can be found in the document "Basic RS485 and RS422 Concepts", available on the website www.novusautomation.com.

The table below helps the connection of the RS485 communication interface connectors.

C				Optional connection which improves the communication performance.	**Terminal 23**
GND					
D0	$\overline{D}$	D-	A	Inverted bidirectional data line.	**Terminal 24**
D1	D	D+	B	Bidirectional data line.	**Terminal 25**

Table 2 – RS485 connections

	**The RS485 port IS NOT ISOLATED from the digital input and output circuits and from the USB port circuit.**

5.1.3 ETHERNET CONNECTION

The Ethernet interface is located on the back of **DigiRail NXprog**, as shown in the **Figure 07**, and enables the communication of the device.

5.2 GALVANIC ISOLATION

DigiRail NXprog has galvanic isolation between functional sections of the internal circuitry to ensure that potential differences between parts of the application system do not interfere with the proper functioning of the device, guaranteeing electrical isolation between them. The galvanic isolation of the **DigiRail NXprog** is shown in the figure below:

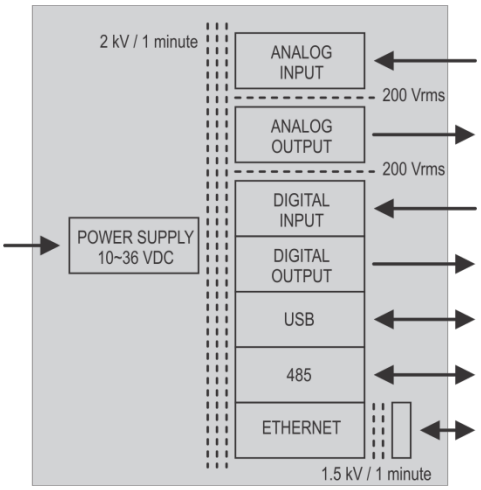

Figure 10 – Galvanic isolation

5.3 ANALOG INPUTS

Located on the frontal panel of **DigiRail NXprog**, the two analog inputs are identified as **A1** and **A2** and are suitable for measuring temperature or any other values represented by standardized linear electrical signals.

Each channel can be configured through the **NXperience** configurator software (see the Configuration Software chapter).

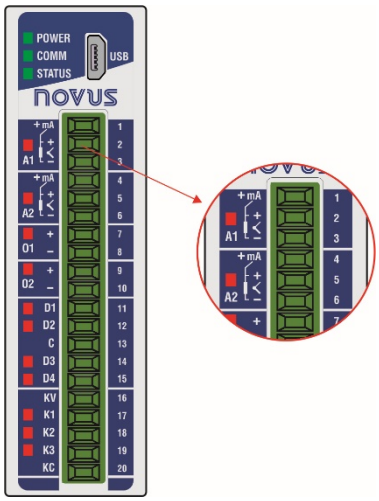

Figure 11 – Analog inputs

The possible input types, along with their respective measuring ranges, are listed in the table below:

TYPE	MEASURING RANGES
J	-110 to 950 °C (-166 to 1742 °F)
K	-150 to 1370 °C (-238 to 2498 °F)
T	-160 to 400 °C (-256 to 752 °F)
N	-270 to 1300 °C (-454 to 2372 °F)
R	-50 to 1760 °C (-58 to 3200 °F)
S	-50 to 1760 °C (-58 to 3200 °F)
B	400 to 1800 °C (752 to 3272 °F)
E	-90 to 730 °C (-130 to 1346 °F)
Pt100	-200 to 850 °C (-328 to 1562 °F)
Pt1000	-200 to 850 °C (-328 to 1562 °F)
NTC	-30 to 120 °C (-22 to 248 °F)
0 – 60 mV	Linear Analog Signals Configurable measuring range: - 65,535 to + 65,535 counts
0 – 5 Vdc	
0 – 10 Vdc	
0 – 20 mA	
4 – 20 mA	

Table 3 – Input types and sensor measurement ranges

Analog input channels **A1** and **A2** are not electrically isolated from each other but are electrically isolated from other **DigiRail NXprog** circuits.

For the **Temperature Sensors** group, the temperature unit setting is required. For the **Linear Analog Signals** group, the measurement range definition is required.

For all types of input signals, it is necessary to set values for the operating parameters of the **DigiRail NXprog** analog input channels (see Analog Inputs section of the Configuration Software chapter):

- **Sampling Rate:** Allows you to set the number of readings performed each second by the analog input channel on the received input signal: 1 reading per second or 10 readings per second.
- **Filter:** Allows you to set the **Time Constant** value of a filter to be applied over the measured input signal. Parameter used to improve the stability of the measured signal. Adjustable between 0 and 1200 seconds.

5.3.1 A1 – A2 STATUS LEDS

When lit, the **A1** and **A2** status LEDs indicate that the respective channel is enabled, not reflecting the condition or value of the signal present at its terminals. In addition to indicating whether a channel is enabled, the LEDs also indicate when there is something wrong on the respective channel.

Improper conditions on the input channels are called "Error Condition" and are showed in specific paragraphs of this manual.

5.3.2 ANALOG INPUT ERROR CONDITION

It is called an "Error Condition" every condition of use or improper operation for the **DigiRail NXprog** input channels. Many of the improper conditions are identified and then signaled by flashing the status LED of the respective channel.

The error conditions of the analog inputs are showed in the table below:

INPUT TYPE	ERROR CONDITIONS
Temperature Sensors	• Measures beyond the limits of the operating range • Open input / open signal
0-20 mA	• Measures above 22 mA (± 0.5 mA) (*)
0-5 / 0-10 V	• Measures above 10% of the upper limit (*) • Reverse polarity
4-20 mA	• Measurements below 3.5 mA (± 0.2 mA) • Measures above 22 mA (± 0.5 mA) • Open input / open signal • Reverse polarity

Table 4 – Analog input error conditions

(*) No error indication when sensor is in open input / open signal.

5.3.3 ANALOG INPUTS CONNECTIONS

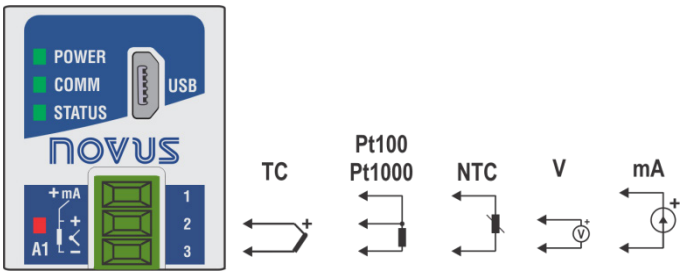

Figure 12 – TC / Pt100 / Pt1000 / NTC / V / mA

5.4 DIGITAL INPUTS

Located on the frontal panel of **DigiRail NXprog**, the four digital inputs are identified as **D1** … **D4** and are suitable for receiving Dry Contact, NPN and PNP electrical signals.

Each channel can be configured through the **NXperience** configurator software (see the Configuration Software chapter).

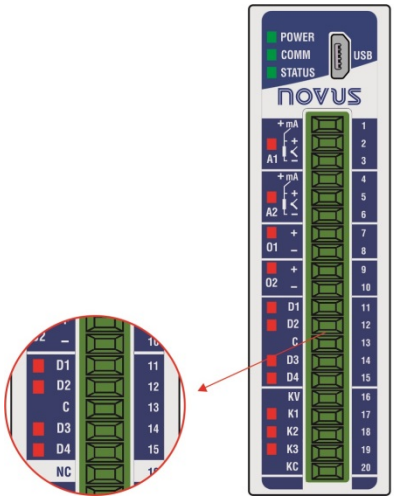

Figure 13 – Digital inputs

The digital inputs can perform different functions, which will be defined during configuration (see Digital Inputs section of the Configuration Software chapter). The functions available are:

- **Logical Status:** When configured as **Logical Status**, allows the device to relate the voltage levels entered into the digital input to the logical status **0** and **1**.
  - **High Logical Level (1):** Voltages higher than 2.2 V.
  - **Low Logical Level (0):** Voltages lower than 1.5 V.
- **Counter:** Allows the digital input to count the number of pulses received at its terminals. As a reference for incrementing the count, you can use the rising edge (transition from **0** to **1**) or the falling edge (transition from **1** to **0**) of the received pulse.
- **Integrator ON/OFF:** Allows the sum (integration) of the time intervals measured with the digital input in logical status **0** to be performed and the sum of the time intervals measured with the digital input in the logical status **1**. It will provide the two information separate. Value displayed in seconds.

In addition, the **Integrator ON/OFF** and the **Counter** functions have the **Preset** function, which allows you to set an initial value for the pulse count or the sum of the digital input ranges to **0** and **1**.

For the **Dry Contact** signal type, there is the **Debounce** feature, which allows defining a time interval to be disregarded by the digital input at each logical status transition.

5.4.1 D1 ... D4 STATUS LEDS

When lit, the **D1** ... **D4** status LEDs indicate the logical status of the signal applied to the terminals of the respective digital input.

5.4.2 DIGITAL INPUTS CONNECTIONS

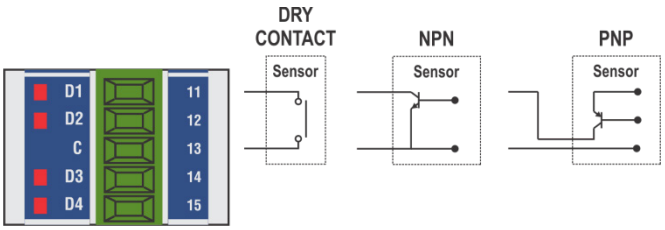

FIGURE 14 – Dry Contact / NPN / PNP

5.5 ANALOG OUTPUTS

Located on the frontal panel of **DigiRail NXprog**, the two analog inputs are identified as **O1** and **O2** and set analog voltage or current values, according to the digital values received.

Each channel can be configured through the **NXperience** configurator software (see the Configuration Software chapter).

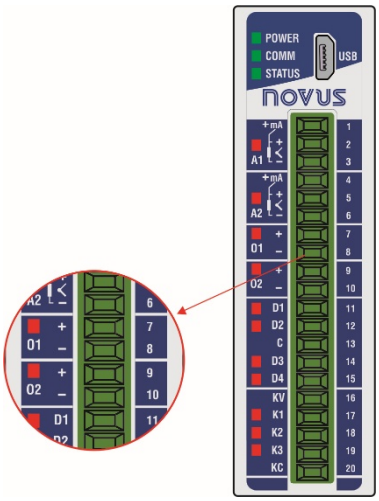

Figure 15 – Analog Outputs

Analog output channels **O1** and **O2** are not electrically isolated from each other but are electrically isolated from other **DigiRail NXprog** circuits.

Analog outputs have three types of signals. There is no need to make any physical changes to the use any of the available output types. Simply set the desired option when configuring **DigiRail NXprog** (see section Analog Outputs in the Configuration Software chapter):

- 0-20 mA;
- 4-20 mA;
- 0-10 V.

After setting the desired output type, you must use the **NXperience** configuration software to set other parameters for operation of the analog outputs in different situations.

- **Operating Range:** Allows you to set the operating range of the analog output:
  - **0.00 to 100.00 %:** The register that controls the analog output expects percentage values within the range 0 to 100 %, where:
    - **0.00 %:** Corresponds to the minimum value of the analog output (0 mA, 4 mA or 0 V);
    - **100.00 %:** Corresponds to the maximum value of the analog output (20 mA, 20 mA or 10 V).
  - **0 to 32 000:** The register that controls the analog output expects the 32 000 size, where:
    - **0:** Corresponds to the minimum value of the analog output (0 mA, 4 mA or 0 V);
    - **32000:** Corresponds to the maximum value of the analog output (20 mA, 20 mA or 10 V).
- **Power On Status:** Allows you to set an initial value for the analog output when turning on the device and receiving a command. There are three possible options:
  - **Disabled:** Allows the analog output to remain off after device initialization and until a valid command is received.
  - **Configured Value:** Allows you to set the value to be adopted in the **Initial Value** parameter after the device initialization and until a valid command is received.
  - **Last Valid Value:** Allows the analog output to adopts the last valid value recorded.
- **Safe Value Watchdog:** Allows you to set a value for the analog output in case of loss of communication.

5.5.1 O1 – O2 STATUS LEDS

When lit, the **O1** and **O2** status LEDs indicate that the respective channel is enabled, not reflecting the condition or value of the signal present at its terminals.

5.5.2 ANALOG OUTPUTS CONNECTIONS

Figure 16 – V / mA

5.6 DIGITAL OUTPUTS

DigiRail NXprog has versions with transistor-sourcing digital output, identified as **K1 ... K4**, and with relay-type digital outputs, identified as **R1 ... R2**, on its front panel.

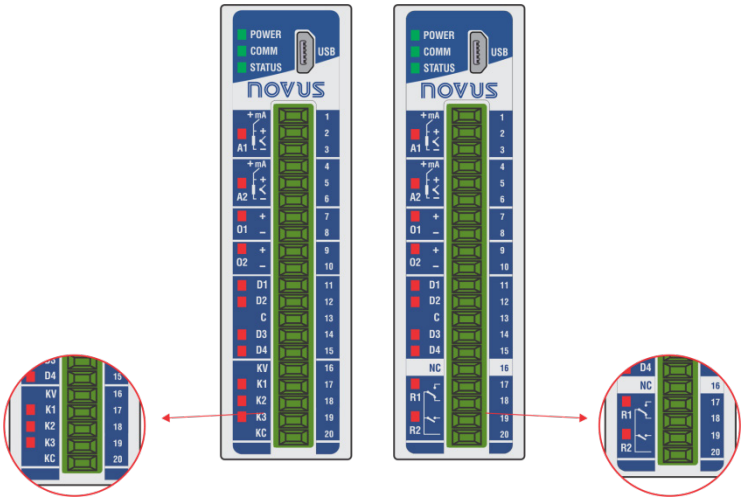

Figure 17 – Digital outputs

DigiRail NXprog has up to four digital outputs (DOs), which obey commands received via digital communication. The registers of the **HR_DOx_VALUE** group are assigned to the digital outputs command. The writing of value **1** in these registers ENABLES the respective digital output. Writing the value **0**, in turn, corresponds to DISABLES the respective digital output.

It is important to note that the ENABLE output status does not necessarily imply that the output is ON or activated.

A group of parameters determines the operation of the digital outputs. These parameters are presented by the **NXperience** configuration software (see Configuration Software chapter), which allows you to define the most suitable configuration for your needs.

The parameters required for configuring the digital outputs are described below.

5.6.1 ACTUATION MODE

The digital output has three actuation modes:

- **Logical Status**: The digital output reproduces the logical status of its respective status register of the **HR_DOx_VALUE** group.

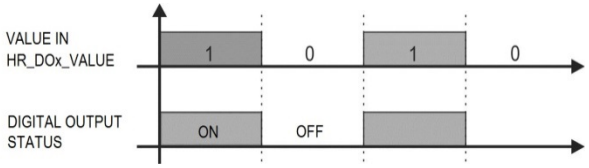

Figure 18 – "Logical Status" mode

- **Pulse**: With the status register receiving the value **1**, the output will turn on for a specific time interval (set in the **Pulse Duration** parameter) and then return to the OFF status.

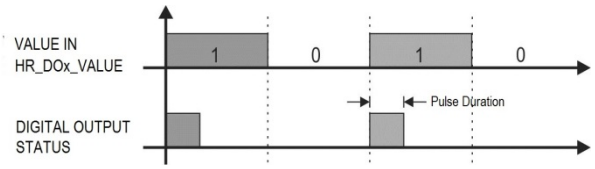

Figure 19 – "Pulse" mode

The ENABLE output status does not necessarily imply that the output is ON or activated.

- **Pulse Train**: With the status register receiving the value **1**, the output will create a defined number of pulses (set in the **Number of Pulses** parameter), with a defined duration (set in the **Pulse Duration** parameter) and in a defined period (set in the **Repetition Period** parameter). After the pulse sequence, the digital output will return to the off status.

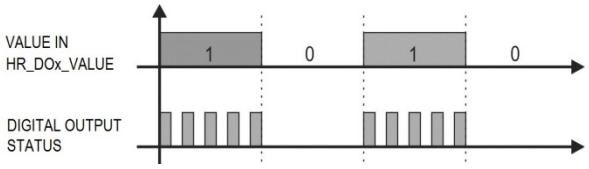

Figure 20 – "Pulse Train" mode

The ENABLE output status does not necessarily imply that the output is ON or activated

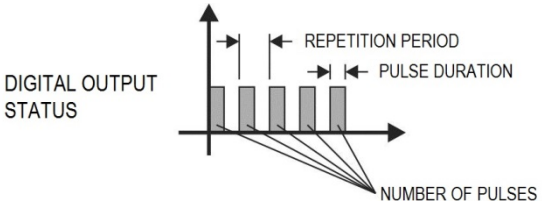

Figure 21 – Digital Output status

5.6.2 POWER ON STATUS

It is the condition to be adopted by the digital output after **DigiRail NXprog** initialization and in which must remain until the receiving of a digital command that redefines its condition. There are three **Power on Status** options:

- **Off:** Allows the digital output to remain off (0) after device initialization.
- **On:** Allows the digital output to start on (1) after device initialization.
- **Last Valid Status:** Allows the digital output to adopt the last valid status registered.

5.6.3 SAFE STATUS WATCHDOG

It allows you to set the condition to be adopted by the digital output when a command is interrupted due to a communication failure.

- **Off:** Allows the digital output to remain off until communication is restored.
- **On:** Allows the digital output to remain on until communication is restored.

5.6.4 K1 ... K4 / R1 ... R2 STATUS LEDS

When lit, the **K1 ... K4** and **R1 ... R2** status LEDs indicate that the respective channel is enabled (logical status **1**).

5.6.5 DIGITAL OUTPUTS CONNECTIONS

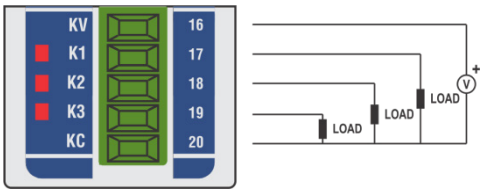

Figure 22 – Digital Outputs (Sourcing)

	**The digital output channels are not electrically isolated from the digital input channels but are isolated from the other DigiRail NXprog electrical circuits.**

5.6.6 RELAY OUTPUTS CONNECTIONS

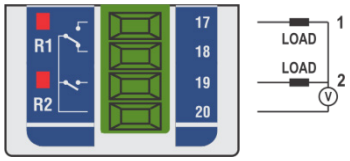

Figure 23 – Relay Outputs

5.7 LEDS

5.7.1 OPERATION LED

It indicates whether or not the device is electrically powered.

5.7.2 RS485 COMMUNICATION LED

If the LED is blinking, it indicates that communication is in progress via the RS485 interface. If the LED is off, it indicates that the device is not communicating via the RS485 interface.

5.7.3 STATUS LED

The LED is controlled by an Arduino application. It is able to assume any function desired by the user.
For more information, consult the programming documentation available on **NOVUS** website at GitHub (https://github.com/NOVUS-Products/DigiRail-NXprog/).

5.7.4 STATUS LED OF EACH CHANNEL

The operation of the status LED of each channel can be viewed in the respective channel section of it within this chapter.

6. MODBUS PROTOCOL

DigiRail NXprog is compatible with the Modbus protocol, a data communication protocol used to connect the device to system control and data acquisition (SCADA).

Operating in slave mode, **DigiRail NXprog** can respond at two Modbus addresses: one with direct access to the input and output module and one used by the Arduino IDE running on the programmable module. The configuration of the Modbus addresses of the input and output module can be performed through **NXperience** (see NXPERIENCE CONFIGURATION SOFTWARE chapter). The Modbus address of the Arduino IDE can be defined in the user application (see ARDUINO IDE chapter). Through the Arduino IDE it is also possible to implement a Modbus Master in the user application, which allows the reading of any device via the RS485 interface.

By developing a Master or a Modbus Slave in the Arduino IDE and loading the program into the programmable module, it is possible to develop any application. However, in a much simpler way, NOVUS has developed a set of registers to simplify the development of the application. For this, the registers memory was parted in two: (i) **Input and output module registers**, which allow direct access to all functionalities available in the **Digirail NXprog** input and output module, such as values of analog input, digital input, digital output, etc.; (ii) **Shared registers**, which allow the Arduino program to share programmable module information with the Modbus world without the need to implement the Modbus protocol in Arduino. Thus, Arduino software only needs to write to the shared area of the input and output module and the information will be available via Modbus RTU and Modbus-TCP. See the available language features and some examples on GitHub.

The commands and Modbus registers supported by input and output module of **DigiRail NXprog** are described below. The commands and Modbus registers supported by programmable module depend on the application developed by the user.

The USB interface is only available for configuration via NXperience and programming via Arduino IDE. Thus, Modbus registers are only available via RS485 or Ethernet.

6.1 COMMANDS

6.1.1 READ COILS – 0x01

This command can be used to read binary values from one or up to a maximum of consecutive registers that are accessible as a bit.

You can use this command to access registers 70 (HR_INPUT1_STATE) through 85 (HR_OUTPUT8_STATE) and registers 500 (HR_DO1_VALUE) through 523 (HR_DO8_FORCE_STATE).

Example of reading the status of inputs 1 to 8 and outputs 1 to 8:

Question	
Field	(hex)
Function	01
Initial Address (High part)	00
Initial Address (Low part)	46
Number to be read (High part)	00
Number to be read (Low part)	10

Answer	
Field	(hex)
Function	01
Number of bytes read	02
Status of inputs 1 to 8	5A
Status of outputs 1 to 8	03

Table 5 – Example of input reading (1)

6.1.2 READ DISCRETE INPUTS – 0x02

This command can be used to read binary values from one or up to a maximum of consecutive registers that are accessible as a bit.

You can use this command to access registers 70 (HR_INPUT1_STATE) through 85 (HR_OUTPUT8_STATE) and registers 500 (HR_DO1_VALUE) through 523 (HR_DO8_FORCE_STATE).

Example of reading the status of inputs 1 to 8 and outputs 1 to 8:

Question	
Field	(hex)
Function	02
Initial Address (High part)	00
Initial Address (Low part)	46
Number to be read (High part)	00
Number to be read (Low part)	10

Answer	
Field	(hex)
Function	02
Number of bytes read	02
Status of inputs 1 to 8	5A
Status of outputs 1 to 8	03

Table 6 – Example of input reading (2)

6.1.3 READ HOLDING REGISTERS – 0x03

This command can be used to read the value of one or up to a maximum of consecutive registers that are accessible as 16 bits.

Example of reading the counter from digital input 1:

Question	
Field	(hex)
Function	03

Answer	
Field	(hex)
Function	03

Initial Address (High part)	00
Initial Address (Low part)	16
Number to be read (High part)	00
Number to be read (Low part)	02

Number of bytes read	04
HR_COUNTER1_LO (High part – 22)	D4
HR_COUNTER1_LO (Low part – 22)	00
HR_COUNTER1_HI (High part – 23)	00
HR_COUNTER1_HI (Low part – 23)	30

Table 7 – Example of digital input counter reading (1)

6.1.4 READ INPUT REGISTERS – 0x04

This command can be used to read the value of one or up to a maximum of consecutive registers that are accessible as 16 bits.

Example of reading the counter from digital input 1:

Question	
Field	(hex)
Function	04
Initial Address (High part)	00
Initial Address (Low part)	16
Number to be read (High part)	00
Number to be read (Low part)	02

Answer	
Field	(hex)
Function	04
Number of bytes read	04
HR_COUNTER1_LO (High part – 22)	D4
HR_COUNTER1_LO (Low part – 22)	00
HR_COUNTER1_HI (High part – 23)	00
HR_COUNTER1_HI (Low part – 23)	30

Table 8 – Example of digital input counter reading (2)

6.1.5 WRITE SINGLE COIL – 0x05

This command can be used to write the binary value of a register that is accessible as a bit. As provided in the Modbus protocol, sending the value 0xFF00 puts the coil register at high level; writing 0x0000, in turn, puts it at low level.

You can use this command to access registers 70 (HR_INPUT1_STATE) through 85 (HR_OUTPUT8_STATE) and registers 500 (HR_DO1_VALUE) through 523 (HR_DO8_FORCE_STATE).

Example of setting output 5 to high level:

Question	
Field	(hex)
Function	03
Address HR_DO4_VALUE (High part)	01
Address HR_DO4_VALUE (Low part)	F8
Value that was sent (High part)	FF
Value that was sent (Low part)	00

Answer	
Field	(hex)
Function	05
Address HR_DO4_VALUE (High part)	01
Address HR_DO4_VALUE (Low part)	F8
Value that was sent (High part)	FF
Value that was sent (Low part)	00

Table 9 – Example of setting output 5 to high level

6.1.6 WRITE SINGLE HOLDING REGISTER – 0x06

This command can be used to write to a register that is accessible as 16 bits.

Example of how to handle analog output 1:

Question	
Field	(hex)
Function	06
Address HR_AO1_VALUE (High part)	02
Address HR_AO1_VALUE (Low part)	0C
Value that was sent (High part)	00
Value that was sent (Low part)	FA

Answer	
Field	(hex)
Function	06
Address HR_AO1_VALUE (High part)	02
Address HR_AO1_VALUE (Low part)	0C
Value that was sent (High part)	00
Value that was sent (Low part)	FA

Table 10 – Example of how to handle analog output 1

6.1.7 WRITE MULTIPLE HOLDING REGISTERS – 0x10

This command can be used to write to multiple registers that are accessible as 16 bits.

Example of how to handle digital outputs 1 and 2:

Question	
Field	(hex)
Function	10
Initial Address (High part)	01

Answer	
Field	(hex)
Function	10
Initial Address (High part)	01

Initial Address (Low part)	F4
Number of registers (High part)	00
Number of registers (Low part)	02
Number of bytes	04
Value sent HR_DO1_VALUE (High part)	00
Value sent HR_DO1_VALUE (Low part)	01
Value sent HR_DO2_VALUE (High part)	00
Value sent HR_DO2_VALUE (Low part)	01

Initial Address (Low part)	F4
Number of registers written (High part)	00
Number of registers written (Low part)	02

Table 11 – Example of how to handle digital outputs 1 and 2

6.1.8 MASK WRITE REGISTER – 0x16

This command can be used to write when using an AND and OR bit mask to a register that is accessible as 16 bits. As provided in the Modbus protocol, the operation performed is:

```
Result = (Content_Register AND Mask_And) OR (Mask_Or AND (NOT Mask_And))
```

Example of the operation performed:

Content_Register	0001 0010
Mask_And	1111 0010
Mask_Or	0010 0101
Result	0001 0111

Table 12 – Example of the operation performed

Example of how to handle digital outputs:

Field	(hex)
Function	16
Address HR_DOALL_VALUE (High part)	02
Address HR_DOALL_VALUE (Low part)	12
Mask AND (High part)	00
Mask AND (Low part)	F2
Mask OR (High part)	00
Mask OR (Low part)	25

Field	(hex)
Function	16
Address HR_DOALL_VALUE (High part)	02
Address HR_DOALL_VALUE (Low part)	12
Mask AND (High part)	00
Mask AND (Low part)	F2
Mask OR (High part)	00
Mask OR (Low part)	25

Table 13 – Example of how to handle digital outputs

6.2 INPUT AND OUTPUT MODULE REGISTERS TABLE

Following is the table of registers supported by the **DigiRail NXprog** input and output module:

* R/W column indicates whether the parameter is writable (R/W) or read-only (R).

** All 32-bit registers consist of two 16-bit registers (LO, HI) and display the information in INT32.

MODBUS ADDRESS	REGISTER	DESCRIPTION	R/W	ACCESS
0	HR_NUM_SERIE_LO	Device serial number (2 registers).	R	16 bits
1	HR_NUM_SERIE_HI			16 bits
2	HR_HW_SET_LO	Hardware configuration. There are two 16-bit registers. Each of the bits represents the presence of a device or channel: 2 HR_HW_SET_LO 3 HR_HW_SET_HI The lower part (HR_HW_SET_LO) is composed of bits from 0 to 15. The upper part (HR_HW_SET_HI) is composed of bits from 16 to 31. **BIT 0:** DI0: Digital Input 1 **BIT 1:** DI1: Digital Input 2 **BIT 2:** DI2: Digital Input 3 **BIT 3:** DI3: Digital Input 4 **BIT 4:** Reserved **BIT 5:** Reserved	R	16 bits
3	HR_HW_SET_HI			16 bits

MODBUS ADDRESS	REGISTER	DESCRIPTION	R/W	ACCESS
		BIT 6: Reserved **BIT 7:** Reserved **BIT 8:** DO0: Digital Output 1 **BIT 9:** DO1: Digital Output 2 **BIT10:** DO2: Digital Output 3 **BIT11:** DO3: Digital Output 4 **BIT12:** Reserved **BIT13:** Reserved **BIT14:** Reserved **BIT15:** Reserved **BIT16:** DO0_Relay: Relay Output 1 **BIT17:** DO1_Relay: Relay Output 2 **BIT18:** Reserved **BIT19:** Reserved **BIT20:** Reserved **BIT21:** Reserved **BIT22:** Reserved **BIT23:** Reserved **BIT24:** AO0: Analog Output 1 **BIT25:** AO1: Analog Output 2 **BIT26:** AI0: Analog Input 1 **BIT27:** AI1: Analog Input 2 **BIT28:** Ethernet: Ethernet Interface **BIT29:** Reserved **BIT30:** RS485: RS485 Interface **BIT31:** Reserved		
4	HR_ETH_MAC0	Ethernet interface MAC address: 6H:6L:5H:5L:4H:4L	R	16 bits
5	HR_ETH_MAC1			16 bits
6	HR_ETH_MAC2			16 bits
7	HR_TS_CALIB0	Date of last calibration (Unix Timestamp, UTC).	R	16 bits
8	HR_TS_CALIB1			16 bits
9	HR_TS_CALIB2			16 bits
10	HR_TS_CALIB3			16 bits
11	HR_VERSAO_FW	Firmware version.	R	16 bits
12	HR_ID	Identification code: 0x0300 (hexadecimal).	R	16 bits
14	HR_AI1_LO	Value read from A1 input. Temperature sensors have a fixed decimal place. Linear sensors respect the range configured by the user.	R	16 bits
15	HR_AI1_HI			16 bits
16	HR_AI2_LO	Value read from A2 input. Temperature sensors have a fixed decimal place. Linear sensors respect the range configured by the user.	R	16 bits
17	HR_AI2_HI			16 bits
18	HR_AO1_LO	Current value of O1 output.	R	16 bits
19	HR_AO1_HI			16 bits
20	HR_AO2_LO	Current value of O2 output.	R	16 bits
21	HR_AO2_HI			16 bits

MODBUS ADDRESS	REGISTER	DESCRIPTION	R/W	ACCESS
22	HR_COUNTER1_LO	Current value from counter of D1 input.	R	16 bits
23	HR_COUNTER1_HI			16 bits
24	HR_COUNTER2_LO	Current value from counter of D2 input.	R	16 bits
25	HR_COUNTER2_HI			16 bits
26	HR_COUNTER3_LO	Current value from counter of D3 input.	R	16 bits
27	HR_COUNTER3_HI			16 bits
28	HR_COUNTER4_LO	Current value from counter of D4 input.	R	16 bits
29	HR_COUNTER4_HI			16 bits
38	HR_DI1_TIME_ON_LO	Current value of time integrator "ON" of D1 input.	R	16 bits
39	HR_DI1_TIME_ON_HI			16 bits
40	HR_DI2_TIME_ON_LO	Current value of time integrator "ON" of D2 input.	R	16 bits
41	HR_DI2_TIME_ON_HI			16 bits
42	HR_DI3_TIME_ON_LO	Current value of time integrator "ON" of D3 input.	R	16 bits
43	HR_DI3_TIME_ON_HI			16 bits
44	HR_DI4_TIME_ON_LO	Current value of time integrator "ON" of D4 input.	R	16 bits
45	HR_DI4_TIME_ON_HI			16 bits
54	HR_DI1_TIME_OFF_LO	Current value of time integrator "OFF" of D1 input.	R	16 bits
55	HR_DI1_TIME_OFF_HI			16 bits
56	HR_DI2_TIME_OFF_LO	Current value of time integrator "OFF" of D2 input.	R	16 bits
57	HR_DI2_TIME_OFF_HI			16 bits
58	HR_DI3_TIME_OFF_LO	Current value of time integrator "OFF" of D3 input.	R	16 bits
59	HR_DI3_TIME_OFF_HI			16 bits
60	HR_DI4_TIME_OFF_LO	Current value of time integrator "OFF" of D4 input.	R	16 bits
61	HR_DI4_TIME_OFF_HI			16 bits
70	HR_INPUT1_STATE	D1 input status.	R	bit / 16 bits
71	HR_INPUT2_STATE	D2 input status.	R	bit / 16 bits
72	HR_INPUT3_STATE	D3 input status.	R	bit / 16 bits
73	HR_INPUT4_STATE	D4 input status.	R	bit / 16 bits
78	HR_OUTPUT1_STATE	K1/R2 output status.	R	bit / 16 bits
79	HR_OUTPUT2_STATE	K2/R2 output status.	R	bit / 16 bits
80	HR_OUTPUT3_STATE	K3 output status.	R	bit / 16 bits
81	HR_OUTPUT4_STATE	K4 output status.	R	bit / 16 bits
94	HR_INTERNAL_TEMP	Cold Junction temperature value. Thus, the Cold Junction will be compensated for thermocouple measurements.	R	16 bits
98	HR_STATUS_AI_CH1	Channel A1 status flags.	R	16 bits
99	HR_STATUS_AI_CH2	Channel A2 status flags.	R	16 bits

MODBUS ADDRESS	REGISTER	DESCRIPTION	R/W	ACCESS
132	HR_INFO_ETH_IPV4_LO	IPv4 Address. Example: IP=192.168.0.1: HR_INFO_ETH_IPV4_HI=0xC0A8 (hexadecimal) HR_INFO_ETH_IPV4_LO=0x0001 (hexadecimal)	R	16 bits
133	HR_INFO_ETH_IPv4_HI			16 bits
134	HR_INFO_ETH_IPV4_SBNT_MSK_LO	IPv4 Subnet Mask (Same IP address format).	R	16 bits
135	HR_INFO_ETH_IPV4_SBNT_MSK_HI			16 bits
136	HR_INFO_ETH_IPV4_DFLT_GTWY_LO	IPv4 Default Gateway (Same IP address format).	R	16 bits
137	HR_INFO_ETH_IPV4_DFLT_GTWY_HI			16 bits
140	HR_TOTAL_SOCKETS	Number of available sockets.	R	16 bits
141	HR_SOCKETS_IN_USE	Number of sockets in use.	R	16 bits
142	HR_GENERAL_ERROR_LO	Ethernet interface error counter.	R	16 bits
143	HR_GENERAL_ERROR_HI			16 bits
144	HR_RELISTEN_ERROR_LO	Relisten error counter.	R	16 bits
145	HR_RELISTEN_ERROR_HI			16 bits
146	HR_SOCKET_SWITCH_ERROR_LO	Counter for socket switching errors.	R	16 bits
147	HR_SOCKET_SWITCH_ERROR_HI			16 bits
148	HR_DISCONNECT_ERROR_LO	Disconnect error counter.	R	16 bits
149	HR_DISCONNECT_ERROR_HI			16 bits
150	HR_SOCKET_CREATION_ERROR_LO	Counter for socket creation errors.	R	16 bits
151	HR_SOCKET_CREATION_ERROR_HI			16 bits
152	HR_SOCKET_DELETE_ERROR_LO	Error counter for socket deletion.	R	16 bits
153	HR_SOCKET_DELETE_ERROR_HI			16 bits
154	HR_IP_INVALID_PACKETS_LO	Number of invalid packets received.	R	16 bits
155	HR_IP_INVALID_PACKETS_HI			16 bits
156	HR_PACKETS_SENT_LO	Number of packets sent.	R	16 bits
157	HR_PACKETS_SENT_HI			16 bits
158	HR_PACKETS_RECEIVED_LO	Number of packets received.	R	16 bits
159	HR_PACKETS_RECEIVED_HI			16 bits
160	HR_ALLINPUTS_STATE	Concatenates the digital inputs status: **BIT 0:** DI0: Digital input 1 **BIT 1:** DI1: Digital input 2 **BIT 2:** DI2: Digital input 3 **BIT 3:** DI3: Digital input 4	R	16 bits
162	HR_ALLOUTPUTS_STATE	Concatenates the digital outputs and relay status: **BIT 0:** DO0: Output K1/R1 **BIT 1:** DO1: Output K2/R2 **BIT 2:** DO2: Output K3 **BIT 3:** DO3: Output K4	R	16 bits
500	HR_DO1_VALUE	Register for handling the Output K1/R1 status.	R/W	16 bits
501	HR_DO2_VALUE	Register for handling the Output K2/R2 status.	R/W	16 bits
502	HR_DO3_VALUE	Register for handling the Output K3 status.	R/W	16 bits

MODBUS ADDRESS	REGISTER	DESCRIPTION	R/W	ACCESS
503	HR_DO4_VALUE	Register for handling the Output K4 status.	R/W	16 bits
508	HR_DO1_STATE_TO_FORCE	Value to force K1/R1 output.	R/W	16 bits
509	HR_DO1_FORCE_STATE	Enables the possibility to force K1/R1 output.	R/W	16 bits
510	HR_DO2_STATE_TO_FORCE	Value to force K2/R2 output.	R/W	16 bits
511	HR_DO2_FORCE_STATE	Enables the possibility to force K2/R2 output.	R/W	16 bits
512	HR_DO3_STATE_TO_FORCE	Value to force K3 output.	R/W	16 bits
513	HR_DO3_FORCE_STATE	Enables the possibility to force K3 output.	R/W	16 bits
514	HR_DO4_STATE_TO_FORCE	Value to force K4 output.	R/W	16 bits
515	HR_DO4_FORCE_STATE	Enables the possibility to force K4 output.	R/W	16 bits
524	HR_AO1_VALUE	Register of manipulation of values applied by output O1.	R/W	16 bits
525	HR_AO2_VALUE	Register of manipulation of values applied by output O2.	R/W	16 bits
526	HR_AO1_VALUE_TO_FORCE	Value to force O1 output.	R/W	16 bits
527	HR_AO1_FORCE_VALUE	Enables the possibility to force O1 output.	R/W	16 bits
528	HR_AO2_VALUE_TO_FORCE	Value to force O2 output.	R/W	16 bits
529	HR_AO2_FORCE_VALUE	Enables the possibility to force O2 output.	R/W	16 bits
530	HR_DOALL_VALUE	Concatenated manipulation register on the digital outputs and relay status (Kn/Rn).	R/W	16 bits
1530	HR_DI1_FORCE_LO	Value to force D1 input (logical status, counter, or time integrator).	R/W	16 bits
1531	HR_DI1_FORCE_HI			16 bits
1533	HR_DI1_FORCE	Enables the possibility to force D1 input.	R/W	16 bits
1580	HR_DI2_FORCE_LO	Value to force D2 input (logical status, counter, or time integrator).	R/W	16 bits
1581	HR_DI2_FORCE_HI			16 bits
1583	HR_DI2_FORCE	Enables the possibility to force D2 input.	R/W	16 bits
1630	HR_DI3_FORCE_LO	Value to force D3 input (logical status, counter, or time integrator).	R/W	16 bits
1631	HR_DI3_FORCE_HI			16 bits
1633	HR_DI3_FORCE	Enables the possibility to force D3 input.	R/W	16 bits
1680	HR_DI4_FORCE_LO	Value to force D4 input (logical status, counter, or time integrator).	R/W	16 bits
1681	HR_DI4_FORCE_HI			16 bits
1683	HR_DI4_FORCE	Enables the possibility to force D4 input.	R/W	16 bits
1880	HR_DI8_FORCE_LO	Value to force D8 input (logical status, counter, or time integrator).	R/W	16 bits
1881	HR_DI8_FORCE_HI			16 bits
1883	HR_DI8_FORCE	Enables the possibility to force D8 input.	R/W	16 bits
2333	HR_AI1_FORCE_VALUE	Enables the possibility to force A1 input.	R/W	16 bits
2334	HR_AI1_FORCED_LO	Value to force A1 input (32 bits).	R/W	16 bits
2335	HR_AI1_FORCED_HI			16 bits
2383	HR_AI2_FORCE_VALUE	Enables the possibility to force A2 input.	R/W	16 bits

MODBUS ADDRESS	REGISTER	DESCRIPTION	R/W	ACCESS
2384	HR_AI2_FORCED_LO	Value to force A2 input (32 bits).	R/W	16 bits
2385	HR_AI2_FORCED_HI			16 bits

Table 14 – Input and output module registers table

6.3 SHARED REGISTERS TABLE

To simplify data sharing of the Arduino application, which runs on the programmable module, the input and output module has 100 registers (from addresses 400 to 499) for free use of the application. In these registers, the Arduino program can read and write freely, interacting with the external world, which can access information via Modbus RTU (via RS485) or Modbus-TCP (via Ethernet).

For more information on how to use this shared registers table, consult the SpecialRegisters page at GitHub (https://github.com/NOVUS-Products/DigiRail-NXprog/blob/master/pages/SpecialRegisters.md).

Following is the table of registers supported by the **DigiRail NXprog** input and output module:

MODBUS ADDRESS	REGISTER	DESCRIPTION
400	HR_APPLICATION_00	Reading and writing registers to use in Arduino application.
...	...	
499	HR_APPLICATION_99	

Table 15 – Shared registers table

7. ARDUINO IDE

The programmable module of **DigiRail NXprog** was designed to allow the user to develop its own application in an easy and intuitive environment. For this, **NOVUS** used the most modern and widespread technology, Arduino IDE, and created libraries to access the hardware.

7.1 INSTALLING NOVUS NXPROG CORE SUPPORT IN ARDUINO IDE

The **NOVUS DigiRail NXprog** Core requires Arduino IDE 1.8.6 or above. To install Arduino IDE, you must follow the steps below:

1) If not installed, download Arduino Desktop IDE.
2) After download, follow the installation guidelines as described in Install the Arduino Desktop IDE link.
3) After installation, run the software and click **File >> Preferences**. A window will appear link the one shown below:

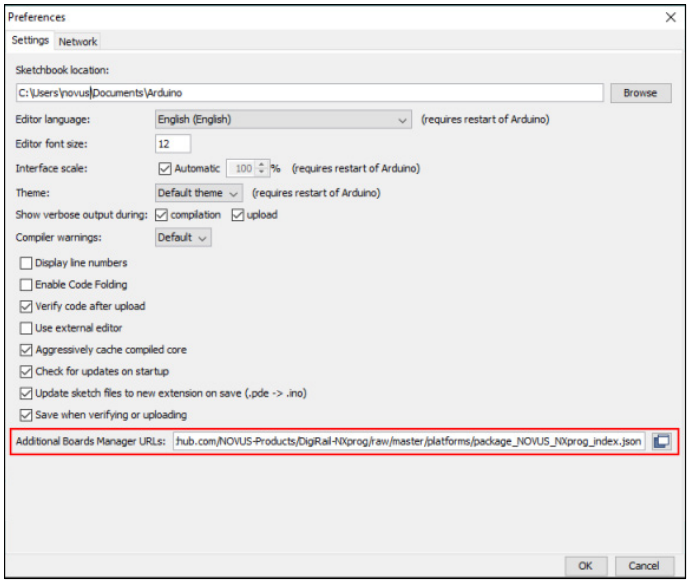

Figure 24 – Arduino IDE installation

4) Click the button next to **Additional Boards Manager URLs**.
5) Add the reference to **NOVUS** platform definition: https://www.novusautomation.com/en/package_NOVUS_NXprog_index.json.
6) Save preferences, then click **Tools**, select the board, and click **Boards Manager**:

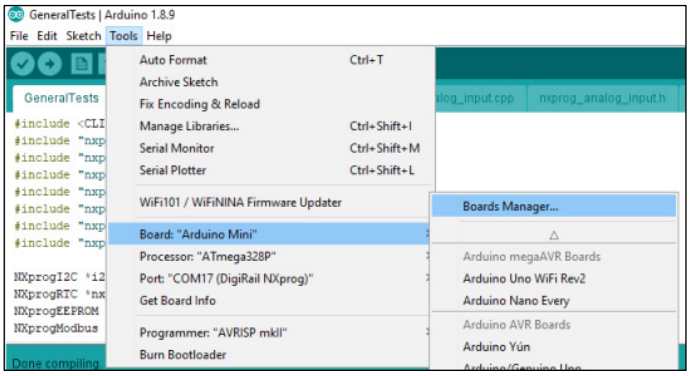

Figure 25 – Board manager

7) Search for **NXprog** in **Boards Manager:**

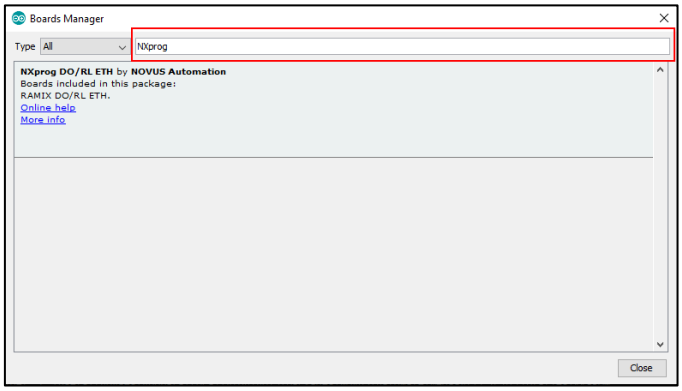

Figure 26 – NXprog

8) Install the NOVUS package.

9) Close **Boards Manager**, then click **Tools >> Board** and select your **DigiRail NXprog** device under **NXprog Arduino Boards** section.

10) Plug in **DigiRail NXprog**.

11) Click **Tools >> Port** and choose the COM port.

12) You can now upload your own sketch.

7.2 LIBRARIES AND EXCLUSIVE FUNCTIONS OF DIGIRAIL NXPROG

DigiRail NXprog programming uses virtually the entire Standard Library defined in the Arduino IDE, which means existing applications can be ported for use on the device. Functions such as *digitalRead* and *analogWrite* are available and require only the correct indication of ports.

The code below, for example, is fully compatible with **DigiRail NXprog**:

```
val = digitalRead(D3);
```

As can be seen above, the reading will be made on the digital port D3. To use the **DigiRail NXprog** ports in the application, just use the nomenclature of the front panel (see the front panel in **Figures 03**, **04**, **05** and **06**).

In addition to the standard functions of the Arduino library, **DigiRail NXprog** offers several libraries and functions to explore the possibilities of the input and output module. An example of this is the various port configuration possibilities presented in the NovusExpert object.

The code below configures analog input A1:

```
NovusExpert.analogInput_setMode(A1, tc_J, CELSIUS, 0);
```

In this example, the analog input is configured to operate with a type J thermocouple, presenting the temperature in degrees Celsius, and will have a value of 0 in case of error.

To explore all the possibilities offered by unique functions of **DigiRail NXprog**, refer to the LANGUAGE REFERENCE section available on the **NOVUS** page on GitHub (https://github.com/NOVUS-Products/DigiRail-NXprog/), which also presents some examples of use for each device function.

8. NXPERIENCE CONFIGURATION SOFTWARE

NXperience software allows you to configure and analyze **DigiRail NXprog** data. With the software, you can explore all the features of the device, communicating through its USB interface.

In addition, **NXperience** allows you to force values into the analog and digital inputs and outputs and perform analysis of information about the Ethernet interface and device status.

NXperience is the most complete configuration tool for the new line of **NOVUS** devices. The software can be downloaded free of charge from our website www.novus.com.br, in the Downloads area.

	**The USB interface powers the device for configuration only and does not allow full use of the device. Thus, the RS485 interface and the analog and digital inputs and outputs will only work when the power supply is connected.**

8.1 CONFIGURING DIGIRAIL NXPROG WITH NXPERIENCE

You can configure **DigiRail NXprog** by clicking the **Configure** button located on the **NXperience** home screen. The following sections describe each of the configuration passable parameters and their particularities.

8.1.1 GENERAL DEVICE INFORMATION

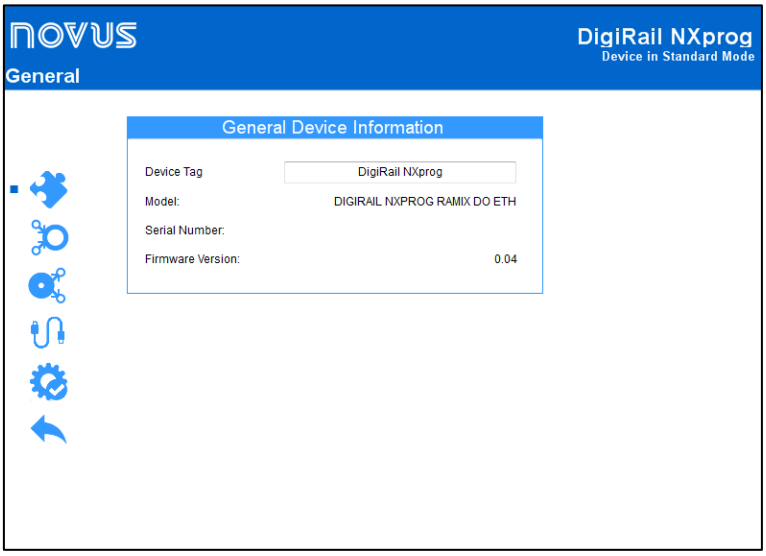

Figure 27 – General parameters

- **Device Tag:** It allows you to set a name, which will be used as an identifier, for the device. The field allows up to 24 characters.
- **Model:** It displays the device model.
- **Serial Number:** It displays the unique device identification number.
- **Firmware Version:** It displays the firmware version recorded on the device.

8.1.2 ANALOG INPUTS

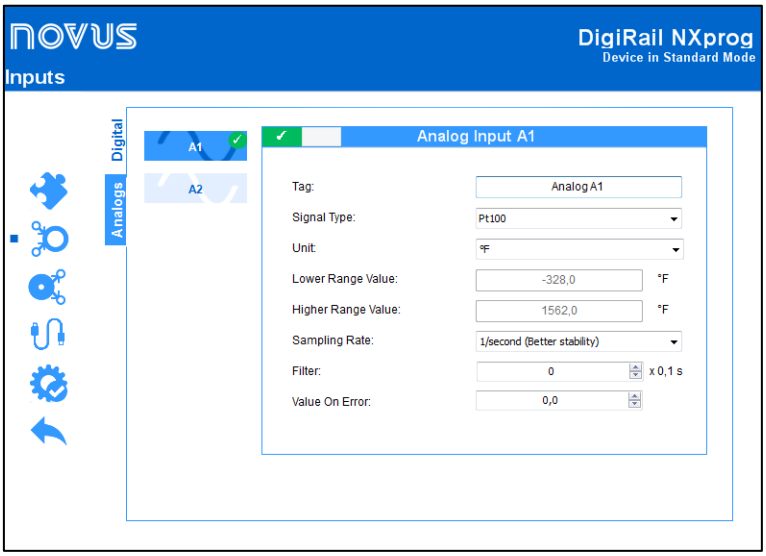

Figure 28 – Analog input

- **Tag:** It allows you to set a name, which will be used as an identifier, for analog input. The field allows up to 24 characters.
- **Signal Type:** It allows you to configure the type of sensor to be used for each analog input.
- **Unit:** It allows you to configure the unit of each analog input. In the case of temperature sensors, it is possible to select the °C or °F units.
- **Lower Range Value:** With lineal signals, it allows you to set a lower value for the range -65.535 to 65.535.
- **Upper Range Value:** With lineal signals, it allows you to set a higher value for the range -65.535 to 65.535.
- **Sampling Rate:** It allows you to set a sampling rate of 1 per second (which gives you better stability) or 10 per second (which gives you worst stability).
- **Filter:** It allows you to define a filter for the selected analog input.
- **Value on Error:** It allows you to define a value to be displayed when there is an error in the configured input.

8.1.3 DIGITAL INPUTS

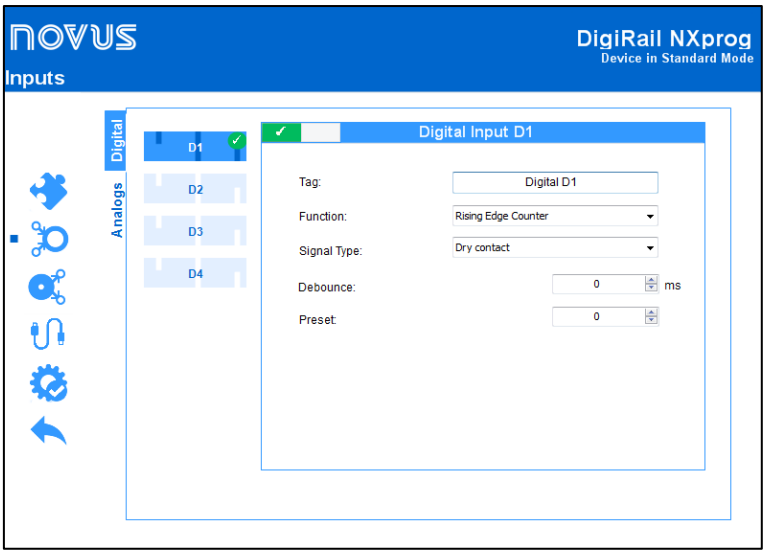

Figure 29 – Digital inputs

- **Tag:** It allows you to set a name, which will be used as an identifier, for digital input. The field allows up to 24 characters.
- **Function:** It allows you to select the function to be performed by the digital input.
  - **Logical Status:** It allows you to read the logical status of the signal applied to the digital input.
    - **High Logical Level (1):** Voltages higher than 2.2 V.
    - **Low Logical Level (0):** Voltages lower than 1.5 V.
  - **Rising Edge Counter:** It allows you to count the number of pulses received at the Rising edge. Up to 250 Hz.
  - **Falling Edge Counter:** It allows you to count the number of pulses received at the Falling Edge. Up to 250 Hz.
  - **Integrator ON/OFF:** It allows the time intervals of the digital input to be integrated into a recorder and, in another register, the time intervals of the connected digital input. Amount accounted in seconds.
- **Signal Type:** It allows you to configure the type of sensor to be used.

- **Debounce:** It allows you to set a time to be disregarded by the counter after detecting the edge at the input. Functionality available when selecting the sensor type Dry Contact. Interval limited to 10 s (10 000 ms) maximum.
- **Preset:** It allows you to set an initial value for the Rising Edge, Falling Edge, and Integrator ON/OFF counters.

8.1.4 ANALOG OUTPUTS

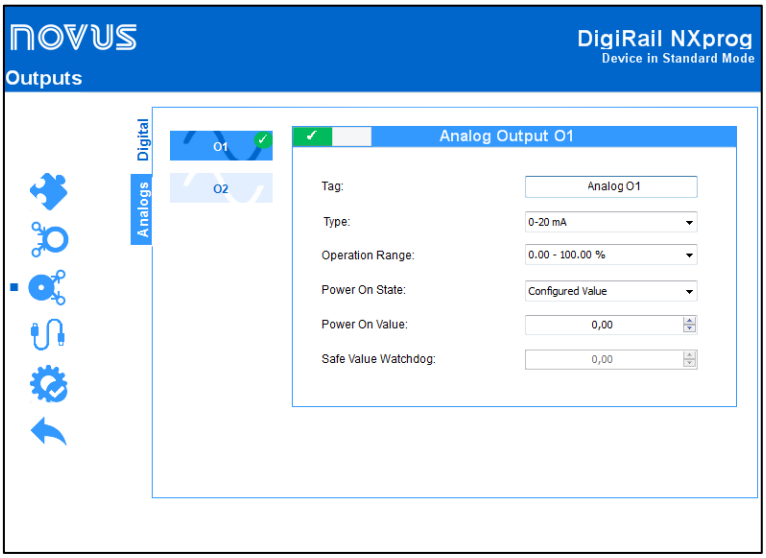

Figure 30 – Analog outputs

- **Tag:** It allows you to set a name, which will be used as an identifier, for the analog output. The field allows up to 24 characters.
- **Operating Range:** It allows you to set the analog output scale, which can be
- **Power On Status:** It allows you to set a value for the analog output when the device is turned on and before an analog output value setting command is received. There are three possible options:
  - **Disabled:** It allows the analog output to remain off after device initialization and until a valid command is received.
  - **Configured Value:** It allows you to set the value to be adopted in the **Initial Value** parameter after the device initialization and until a valid command is received.
    - **Power On Value:** It allows you to set the value to be adopted by the analog output after the device initialization. This parameter is directly related to the chosen operating range and can be any value within the range 0 to 100 % or 0 to 32,000 counts.
  - **Last Valid Value:** It allows determining that, after the device initialization, the analog output adopts the last valid value applied from the respective value manipulation registers provided by (HR_AO "x"_VALUE).
- **Safe Value Watchdog:** It allows you to set a value to be adopted by the analog output in case of loss of Ethernet and/or RS485 communication.

8.1.5 DIGITAL OUTPUTS

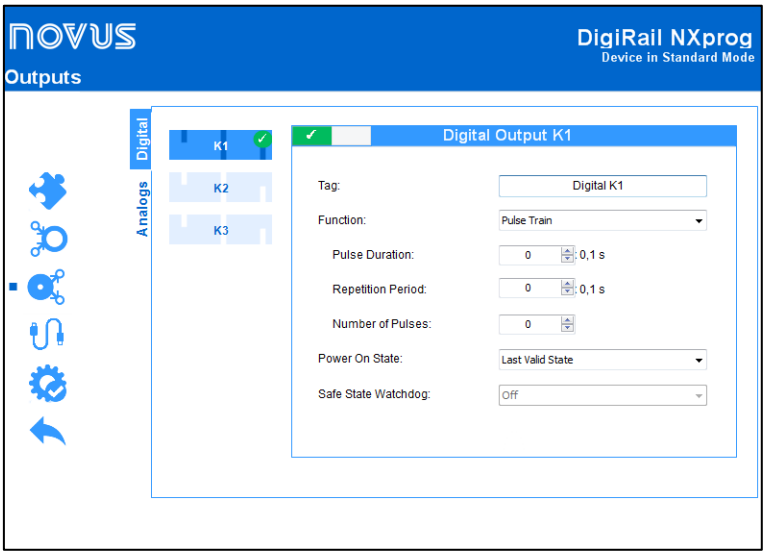

Figure 31 – Digital outputs

- **Tag:** It allows you to set a name, which will be used as an identifier, for the digital output. The field allows up to 24 characters.
- **Actuation Mode:** It allows you to define the mode of operation of the digital output.
  - **Logical Status:** When selected, it allows the value 0 or 1 to be applied to the configured digital output.
  - **Pulse:** When selected, it allows the digital output to be turned on for a set time (defined in the **Pulse Duration** parameter) and after that period returns to the off condition.
    - **Pulse Duration:** It allows you to set the pulse duration and how long the digital output will remain on.
  - **Pulse Train:** When selected, it allows the digital output to generate a defined pulse sequence.
    - **Pulse Duration:** It allows you to set the pulse duration and how long the digital output will remain on.
    - **Repetition Period:** It allows you to define the repetition period of the pulse train, which consists of the interval between the pulses.
    - **Number of Pulses:** It allows you to set the number of pulses to be applied in the configured range.
- **Power On Status:** It allows you to set the initial status of the devices analog output after initializing the device until a command is acknowledged.
  - **Off:** It allows the digital output to remain off (0) after device initialization.
  - **On:** It allows the digital output to start on (1) after device initialization.
  - **Last Valid Status:** It allows the digital output to adopt the last valid status registered.
- **Safe Status Watchdog:** It allows you to set the condition to be adopted by the digital output when a command is interrupted due to a communication failure.
  - **Off:** It allows the digital output to remain off until communication is restored.
  - **On:** It allows the digital output to remain on until communication is restored.

8.1.6 COMMUNICATION PARAMETERS

8.1.6.1 ETHERNET

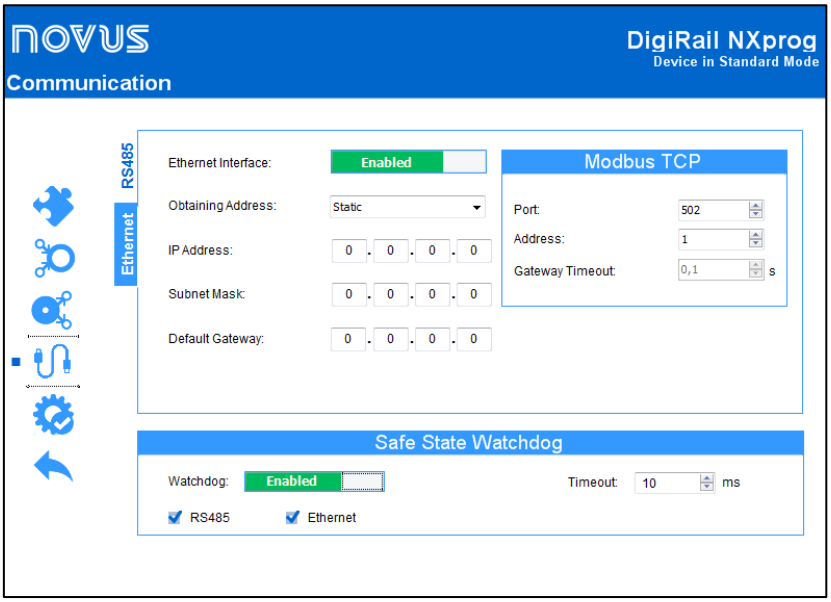

Figure 32 – Communication: Ethernet

INTERNET INTERFACE

- **Ethernet:** It allows you to enable or disable the Ethernet interface.
- **Obtaining Address:** It allows you to define how **DigiRail NXprog** will obtain an IP: **DHCP** (Dynamic Host Configuration Protocol), which allows the IP (Internet Protocol) of the device to be assigned by the network server, or **Static**, which allows the user sets the IP address, subnet mask, and default gateway for the connection.
- **IP Address:** It allows you to enter the IP, which refers to the identification of the device in a local or public network, to be used by the device. This is a required field when the **Obtaining Address** parameter is marked **Static**.
- **Subnet Mask:** Also known as subnet mask or netmask, it allows you to divide a specific network into smaller subnets, making it more effective to use a certain IP address space. This is a required field when the **Obtaining Address** parameter is marked **Static**.
- **Default Gateway:** It allows you to enter a default gateway, which refers to the device address on the network that connects your computers to the Internet, to the device. This is a required field when the **Obtaining Address** parameter is marked **Static**.

MODBUS-TCP

- **Port:** It allows you to define the TCP port on which the service will be available.
- **Address:** It allows you to set the Modbus address to be adopted by the device, so that it can communicate on a Modbus network.
- **Gateway Timeout:** It allows you to set the timeout (in milliseconds) of the gateway. This is a required field when the **Modbus Operation Mode** parameter of the **RS485** tab (see section RS485) is selected as the **Gateway**.

8.1.6.2 RS485

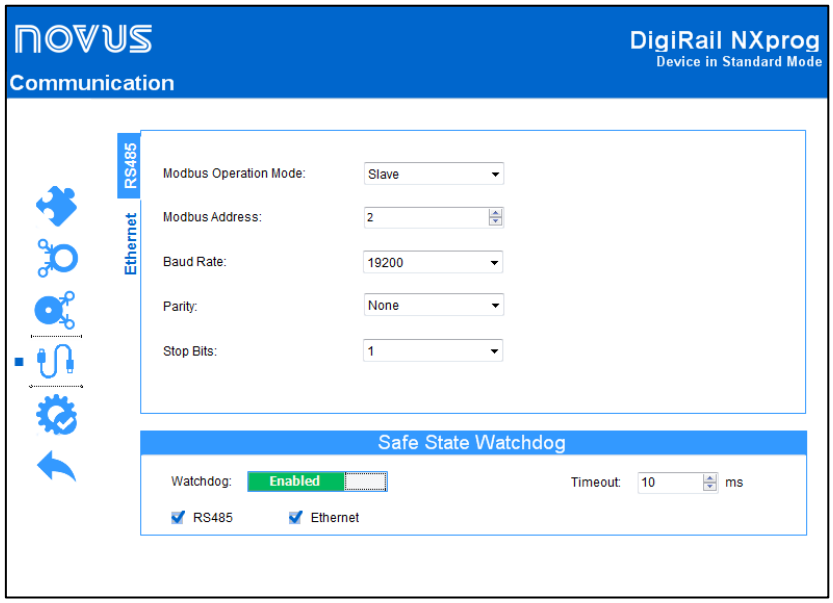

Figure 33 – Communication: RS485

- **Modbus Operation Mode:** It allows you to set the Modbus operation mode of the RS485 interface: **Slave** or **Gateway**.
- **Modbus Address:** It allows you to set the Modbus address to be used by the device, so that it can communicate on a Modbus network. This is an editable field when the **Modbus Operation Mode** parameter is selected as **Slave**. Allows an address between 1 and 247.
- **Baud Rate:** It allows you to set the Baud Rate to be used by the Modbus network.
- **Parity:** It allows you to set the parity to be used by the Modbus network: even, odd or none.
- **Stop Bits:** It allows you to set the number of Stop Bits to be used by the Modbus network.

GATEWAY MODE OPERATION

You can configure **DigiRail NXprog** to operate in Gateway mode between a Modbus-TCP network and a Modbus RTU network. **DigiRail NXprog** will allow a Modbus-TCP client on the Modbus-TCP network (a PLC or a SCADA system via **Ethernet**, for example) to communicate with devices from a Modbus RTU network in RS485. In this mode, the Arduino module cannot operate as Modbus RTU master or slave, because the RS485 interface will be dedicate to access via Modbus-TCP. However, every Arduino application can share information through the shared registers (see Shared Registers Table section).

Along with the Ethernet interface configuration, there is a configuration of Modbus-TCP where, in addition to the port, you can configure the **DigiRail NXprog** RTU address and, if it is enabled, you can also configure the Timeout of the Modbus TCP/RTU gateway. All Modbus-TCP requests received by **DigiRail NXprog** with a different Modbus RTU address than the address configured in the device will be converted to the Modbus RTU protocol and retransmitted in the RS485 network. Responses to these requests will be reconverted to the Modbus-TCP protocol and relayed over the Ethernet network to the requesting Modbus-TCP client.

The **Gateway** mode adapts the protocol to the physical environment and is transparent to the Modbus-TCP client.

8.1.6.3 SAFE STATUS WATCHDOG

- **Watchdog:** It allows you to enable or disable the Watchdog function.
- **Timeout:** It allows you to enter a period of time (in ms) to activate the Watchdog function. If there is a loss of communication and once the timeout time set in this parameter has passed, the analogue or digital output will receive the value previously set in the **Safe Status Watchdog** parameter. This parameter can be configured with a minimum value of 10 ms and a maximum value of 65535 ms.
- **RS485:** If selected, it allows the Watchdog function to act on the RS485 interface.
- **Ethernet:** If selected, it allows the Watchdog function to act on the Ethernet interface.

8.2 DIAGNOSTICS

You can view the **DigiRail NXprog** diagnostics tab by clicking the **Diagnostics** button located on the **NXperience** home screen. In addition to providing an analysis of the communication status of the device, this tab also allows you to force specific values for each channel.

The following sections describe each of the configuration passable parameters and their particularities.

8.2.1 FORCING DIGITAL INPUTS

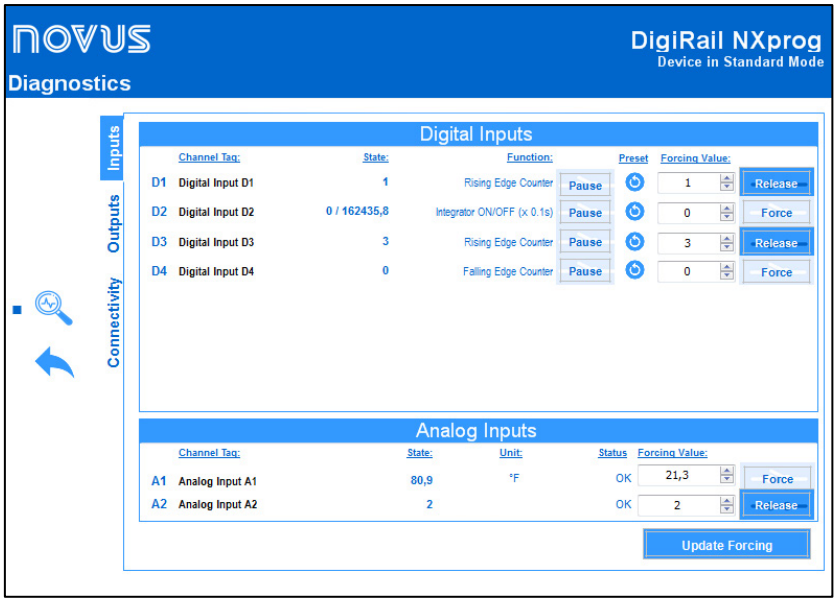

Figure 34 – Diagnostics: Inputs

- **Channel Tag:** It displays the channel tag, defined in the **Tag** parameter of each digital input.
- **Status:** It displays the current value displayed by the channel.
- **Function:** It displays the function configured for each channel, defined in the **Function** parameter of each digital input.
  - **Pause:** Available for **Counter** and **Integrator** functions. It allows pausing the counter/integrator value. This button is used to pause and to resume the counter/integrator.
- **Preset:** It allows you, when clicking the button, to apply to the channel the value previously configured in the **Preset** parameter of the digital input.
- **Forcing Value:** It allows you to force a specific value for each channel by entering the desired value and clicking the Force button. When executing this function, the **Status** parameter will adopt the forced value. To stop forcing, just click the Release button.
- **Update Forcing:** It allows you to update the forcing values of the already forced channels by clicking the respective button.

8.2.2 FORCING ANALOG INPUTS

- **Channel Tag:** It displays the channel tag, defined in the **Tag** parameter of each analog output.
- **Status:** It displays the current value displayed by the channel.
- **Unit:** It displays the unit configured for each analog input, set in the **Unit** parameter of each analog input.
- **Status:** It displays the status of each analog input. **OK** means there is no error in the analog input. **NOK** means there is an error in the analog input.
- **Forcing Value:** It allows you to force a specific value for each channel by entering the desired value and clicking the Force button. When executing this function, the **Status** parameter will adopt the forced value. To stop forcing, just click the Release button.
- **Update Forcing:** It allows you to update the forcing values of the already forced channels by clicking the respective button.

8.2.3 FORCING DIGITAL OUTPUTS

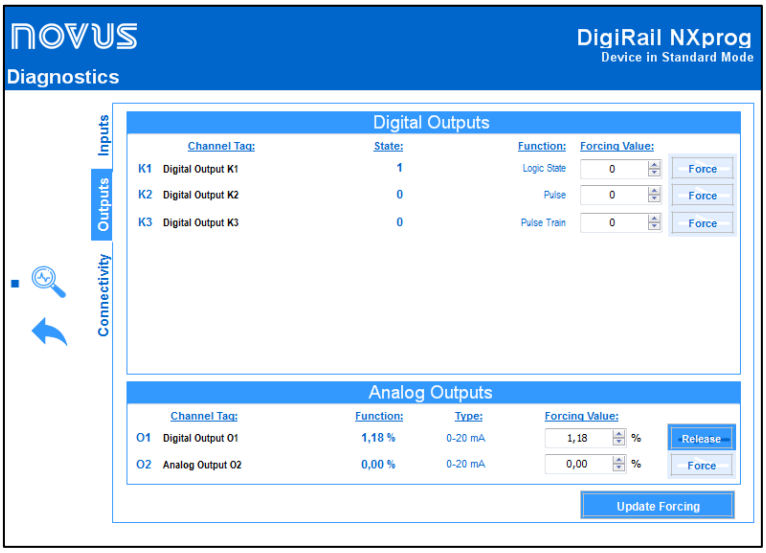

Figure 35 – Diagnostics: Outputs

- **Channel Tag:** It displays the channel tag, defined in the **Tag** parameter of each digital output.
- **Status:** It displays the current value displayed by the channel.
- **Function:** It displays the function configured for each channel, defined in the **Actuation Mode** parameter of each digital output.
- **Forcing Value:** It allows you to force a specific value for each channel by entering the desired value and clicking the Force button. When executing this function, the **Status** parameter will adopt the forced value. To stop forcing, just click the Release button.
- **Update Forcing:** It allows you to update the forcing values of the already forced channels by clicking the respective button.

8.2.4 FORCING ANALOG OUTPUTS

- **Channel Tag:** It displays the channel tag, defined in the **Tag** parameter of each analog output.
- **Status:** It displays the current value displayed by the channel.
- **Type:** It displays the type of output signal configured for each channel: 0-20 mA, 4-20 mA or 0-10 V.
- **Forcing Value:** It allows you to force a specific value for each channel by entering the desired value and clicking the Force button. When executing this function, the **Status** parameter will adopt the forced value. To stop forcing, just click the Release button.
- **Update Forcing:** It allows you to update the forcing values of the already forced channels by clicking the respective button.

8.2.5 COMMUNICATION

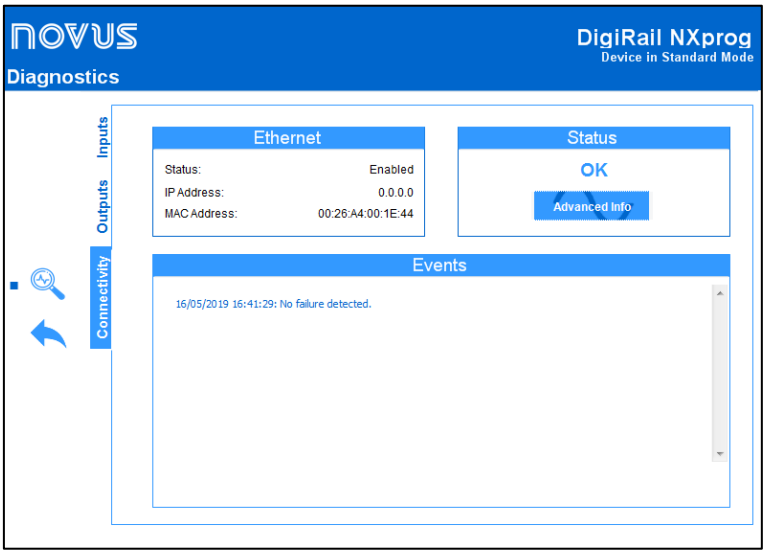

Figure 36 – Diagnostics: Communication

8.2.5.1 ETHERNET

- **Status:** It informs whether the Ethernet interface is enabled.
- **IP Address:** It displays the IP address of the device.
- **MAC Address:** It displays the MAC address of the device.

8.2.5.2 STATUS

It informs if the device is in an error status.

- **Advanced Info:** It displays the date, time and name of the event or error displayed by the device. Check **Table 07** and **Table 08** for further information on this.

EVENT FLAGS
Ethernet communication is enabled and running.
This device is connected via USB.
A Watchdog event occurred on the RS485 interface.
A Watchdog event occurred on the Ethernet interface.
One of the enabled channels has been forced.
One of the enabled channels has been paused.

Table 16 – Event flags

ERROR FLAGS
Analog output failure.
Digital output failure.
Power supply failure.
Analog inputs failure.

Table 17 – Error flags

9. TECHNICAL SPECIFICATION

FEATURES	DIGIRAIL NXPROG	
Input Channels	4 Digitals and 2 Analogs	
Output Channels	3 Digitals or 2 Relays and 2 Analogs	
Analog Input	**Analog Signals Accepted**	Thermocouples J, K, T, N, E, R, S e B, Pt100, Pt1000, NTC, 0-60 mV, 0-5 Vdc, 0-10 Vdc, 0-20 mA, 4-20 mA
	Accuracy of Measure	Thermocouples: 0.2 % of maximum range Pt100, Pt1000, NTC, mA, V, mV: 0.15 % of the maximum range Cold junction error to be considered for measurements with thermo-couples: J, K, T: ± 1 °C / ± 1.8 °F N, E, R, S, B: ± 3 °C / ± 5.4 °F
	Input Impedance from Analog Channels	0-60 mV, Pt100, Pt1000, NTC and thermocouples : >10 MΩ 0-5 V, 0-10 V: >1 MΩ 4-20 mA: 15 Ω + (1 V @ 20 mA)
	Pt100	Maximum compensated cable resistance: 20 Ω Excitation current: 0.60 mA
	Analogue Channel resolution	Analog inputs: 16 bits (65536 levels)
Digital Input	**Logical Levels**	Logical Level "0": < 0.5 V Logical Level "1": > 3 V
	Maximum Voltage	30 V
	Input Impedance	270 kΩ
	Input Current @ 30 Vdc (typi-cal)	0.15 mA
	Maximum Frequency (square wave)	Dry Contact: 10 Hz PNP: 250 Hz NPN: 250 Hz
	Minimum Pulse Duration	Dry Contact: 50 ms PNP: 4 ms NPN: 4 ms
Transistor Digital Output	Transistor outputs (Sourcing) Maximum load current: 500 mA Short-circuit current: 70 mA Maximum switching voltage: 30 Vdc Minimum switching voltage: 6 Vdc	
Relay Digital Output	Type: SPST-NO and SPDT Maximum load current: 3 A (SPST) / 6 A (SPDT) Switching voltage: 250 Vac Suitable for resistive loads	
Analog Output	Signal types: 0-20 mA, 4-20 mA, 0-10 V Maximum load: 0-20 / 4-20 mA: ≤ 500 Ohms 0-10 V: ≥ 2000 Ohms Resolution: 12 bits Accuracy: 0.5 %	
Programmable Module	Programmable Environment	Programmable in Arduino IDE. Support for the standard Arduino library with **NOVUS** extensions.
	Processor	ATMEGA4809 with 48 kB Flash, 6 kB SRAM and 256 B EEPROM.
	RTC	Real time clock with accuracy of +- 3 ppm. Internal backup battery with up to five-year estimated life. 512 B SRAM battery-powered memory.
	EEPROM	256 kb of EEPROM memory available for data storage.
	WDT	Watchdog Timer for monitoring of Arduino code execution.
	BOD	Brown-Out Detection for processor power supply monitoring.
Communication Interfaces	USB Ethernet: 10/100 Mb/s, IEEE standard 802.3u	

	RS485
Configurator Software	**NXperience** (via USB for desktops and notebooks)
Power Supply	Voltage: 10 Vdc to 36 Vdc Maximum consumption: 5 W Typical consumption: 20 mA
Dielectric Rigidity	See **Figure 10**
Operating Temperature and Humidity	Temperature: -20 to 60 °C Humidity: 5 to 95 % RH, non-condensing
Housing	ABS+PC
Protection Rating	IP2O
Dimensions	100 x 30 x 110 mm
Certifications	CE and UKCA

Table 18 – Technical Specification

10. WARRANTY

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