pizzato

Gemnis Studio 12

USER MANUAL

release 12.1.9.0

Help Gemnis Studio
Release 12.1.9.0

1 Introduction 6
2 General characteristics 6
3 Symbols, application limits and limits of liability 7
3.1 Graphic symbols and terms used in this manual 7
3.2 Application limits 7
3.3 Prerequisites 8
3.4 Intended use 8
3.5 User's responsibility 8
3.6 Management of damaged or faulty products 9
3.7 Validation and periodic tests 9
3.8 Directives and standards 10
3.9 Limits of liability 11
4 Gemnis line modules 11
4.1 Introduction 11
4.2 General hardware characteristics 11
4.3 Detection principles of external inputs and faults 13
4.3.1 General principles for external fault detection 13
4.3.1.1 About short-circuit between different inputs. 14
4.3.2 Safety devices 15
4.3.3 Sensors 16
4.4 Modules state 17
4.4.1 POWER-ON State 17
4.4.2 START State 17
4.4.3 SET State 17
4.4.4 RUN State 18
4.4.5 ERROR State 18
4.5 Module state display by P1 and P2 status LEDs 19
4.6 List of products 20
4.6.1 USB Port 21
5 Gemnis Studio 21
5.1 Prerequisites for performing of Gemnis Studio 21
5.2 Software version 21
5.3 Introduction 21
5.3.1 Main menu 22
5.4 Tools Menu 22
5.4.1 General options 22
5.4.1.1 Theme and Language 23
5.4.1.2 Default Desktop Settings. 23
5.4.1.3 Default Grid Settings 23
5.4.1.4 Other 24
5.4.2 Programming by rows of one or more modules 25
5.4.3 Extracting a program from a module 24
5.5 Modules Menu 25
5.6 Projects Menu 26
5.6.1 Project Tab 26
5.6.2 Application program and digital signature 27
5.6.3 Password 29
5.6.4 Migration of programs 31
5.7 User Program 32

 2024 © Pizzato Elettrica S.r.l.

5.7.1 Desktop 32
5.7.1.1 Analysis and validation of the project 37
5.7.2 Sensors 39
5.7.2.1 S1C Sensor 46
5.7.2.2 S1E Sensor 47
5.7.2.3 S2CI Sensor 48
5.7.2.4 S2CD Sensor 49
5.7.2.5 S2ED Sensor 50
5.7.2.6 SSM Sensor 51
5.7.2.7 STHC Sensor 52
5.7.2.8 STHA Sensor 54
5.7.2.9 SSE8 Sensor 55
5.7.2.10 SPE3 Sensor 56
5.7.2.11 SUSB Sensor 57
5.7.2.12 STIN Sensor 59
5.7.2.13 SAN1 Sensor 60
5.7.2.14 SAN3 Sensor 64
5.7.2.15 SSP2 Sensor 66
5.7.2.16 SSP4 Sensor 71
5.7.3 Functional blocks 74
5.7.3.1 Functional block AND 77
5.7.3.2 Functional block OR 77
5.7.3.3 Functional block XOR 78
5.7.3.4 Functional block NAND 78
5.7.3.5 Functional block NOR 79
5.7.3.6 Functional block NXOR 79
5.7.3.7 Functional block NOT 80
5.7.3.8 Functional block TRUE 80
5.7.3.9 Functional block FALSE 80
5.7.3.10 Functional block SET/RESET 81
5.7.3.11 Functional block MEM 81
5.7.3.12 Functional block START 82
5.7.3.13 Functional block POWER ON 84
5.7.3.14 Functional block TRIGGER 84
5.7.3.15 Functional block DELAY 85
5.7.3.16 Functional block PULSE 86
5.7.3.17 Functional block CLOCK 87
5.7.3.18 Functional block FILTER 88
5.7.3.19 Functional block ERROR 89
5.7.3.20 Functional block MESSAGE 91
5.7.3.21 Functional block EQU/GEQ/LEQ 92
5.7.3.22 Functional block COUNTER 93
5.7.3.23 Functional block LKTBL 93
5.7.3.24 Functional block LDC (LOCK DOOR CONTROL) 94
5.7.3.25 Functional Block WAVE 99
5.7.3.26 Functional Block MUTE2 99
5.7.3.27 Functional Block EDM 101
5.7.3.28 Functional Block SERIAL 103
5.7.3.29 Functional block BTST 107
5.7.3.30 Functional block ADIFF 108
5.7.3.31 Functional block AVG 109
5.7.3.32 Functional block WTOB 110

 2024 © Pizzato Elettrica S.r.l.

5.7.3.33 Functional block SUM 111
5.7.3.34 Functional block MUL 112
5.7.4 Outputs 116
5.7.4.1 Safety Outputs 116
5.7.4.2 Test pulses 117
5.7.4.3 Signal outputs (unsafe) 118
5.7.5 Connectors 119
5.7.5.1 Connector controlled by User Dump 122
5.7.6 Other items 123
5.7.6.1 Note 123
5.7.6.2 Labels 124
5.7.6.3 Display 126
5.7.7 Simulation 128
5.7.7.1 Time in simulation 130
5.7.7.2 Start conditions and special states of sensors 130
5.7.7.3 Simulated ERROR conditions 131
5.8 Monitor 131
5.9 Prints 135
5.9.1 Validation Report 135
5.9.2 User Program Report 136
6 System commands 137
6.1 Introduction 137
6.1.1 System Commands format and limitations of use 137
6.1.2 Commands executable in RUN state 138
6.1.3 Commands executable in SET state 138
6.1.4 Commands executable in ERROR state 138
6.2 System Commands list 139
6.2.1 P00 Command 139
6.2.2 P01 Command 139
6.2.3 P03 Command 139
6.2.4 P04 Command 140
6.2.5 P06 Command 140
6.2.6 P07 Command (UserDump) 140
6.3 Writing data into the module 141
6.3.1 WD Command 141
6.3.2 W? Command 142
6.3.3 WK Command 142
6.3.4 WE Command 142
6.3.5 Example of transaction 143
7 Support 144
7.1 Website 144
7.2 Technical support 144
7.3 Making of pre-programmed modules, series CS MF 145

 2024 © Pizzato Elettrica S.r.l.

1 Introduction

Thank you for choosing a product of Pizzato Elettrica Gemnis series. This manual has been developed with the aim to introduce you to the features and details of the programming of this modern and comprehensive family of programmable safety modules.

If you would like more information or if the information you sought were not present in this manual you can connect to the website www.gemnis.com or contact the support service (see 7).

2 General characteristics

A Gemnis series module is a programmable safety device that allows you to perform multiple safety functions at the same time. This product has been specifically developed to meet the needs of manufacturers of machinery having a medium/low number of safety functions. These modules are able to manage minimum applications comparable approximately to the ones performed by 3/4 traditional electromechanical safety modules up to circuits having a few dozen inputs. Since this family of products provides several combinations, in case of doubt please contact support.

These modules are able to manage inputs coming from all of the following types of safety devices:

- Mechanical safety switches;
- Safety switches with solenoid for locking doors;
- Safety magnetic switches;
- Safety light barriers (category 4);
- Safety sensors;
- Mushroom pushbuttons for emergency stop;
- Rope switches for emergency stops;
- Safety mats or safety edges with 4 wires technology;
- Category IIIA or IIIC two hand controls;
- Safety selectors;
- Enabling three states devices;
- Analog sensors with 4-20 mA output;
- Speed detection systems.

They also have internal capabilities that allow to carry out also:

- Safety timing;
- Detection of different types of failures of safety devices or their wirings;
- Check temperature limits internal to the module;
- Math functions like sum, absolute difference, multiplication.

Finally, Gemnis series modules are able to:

- Manage up to eight electronic safety outputs or up to four relay outputs;
- Manage many signal outputs (unsafe);

 2024 © Pizzato Elettrica S.r.l.

- Perform communications of the state of the module and data settings via the built-in USB port.

Safety modules project Gemnis are able to realize safety circuits up to “Maximum SIL” 3 classification according to EN IEC 62061, PL “e” and Category 4 according to EN ISO 13849-1.

3 Symbols, application limits and limits of liability

3.1 Graphic symbols and terms used in this manual

This manual contains some graphic symbols whose meaning is as follows:

⚠ This symbol indicates that there are very important information or requirements for safety. Failure to follow these indications may represent a serious danger to users of machinery.

ⓘ This symbol indicates additional information useful for a better understanding of what was reported.

Text This formatting indicates a button or a menu item of the software Gemnis Studio

Terms	See
Functional blocks	5.7.3
Connectors	5.7.5
Safety devices	4.3.2
MTTFd (Mean Time To dangerous Failure)	See standards EN ISO 13849-1, EN IEC 62061
PFHd (Probability of dangerous Failure per Hour)	See standards EN ISO 13849-1, EN IEC 62061
PL (Performance Level)	See EN ISO 13849-1
Projects	5.6
Application Program	5.6.2
Sensors	4.3.3 and 5.7.2
SIL (Safety Integrity Level)	See EN IEC 62061
Maximum SIL (Safety Integrity Level)	See EN IEC 62061
Module state	4.4

3.2 Application limits

⚠ A Gemnis safety module can be used to create safety circuits up to “Maximum SIL” 3 according to EN IEC 62061 or PL “e” according to EN ISO 13849-1. The achievement of a specific safety level for a given circuit is not only determined by the module, but also by external components connected to it, by the circuit structure, and by a number of additional requirements as required by the standards mentioned above.

2024 © Pizzato Elettrica S.r.l.

3.3 Prerequisites

⚠ In order to create a program correctly, intended for a Gemnis series module to carry out safety functions, a variety of skills is required, in particular:

- The knowledge of the used module and its application limits as prescribed in its instruction manual, as well as knowledge of devices applied to it (switches, etc.) and their limits.
- Knowledge of configuration software (Gemnis Studio) and the requirements set out in this manual.
- Knowledge of European Directives applicable to machinery on which you will install the safety circuit governed by the module and consequently the knowledge of specific standards necessary to make the risk analysis of the machinery, and standards needed to select the safety measures designed to prevent or mitigate the risks identified.

The lack of knowledge of the limitations of devices employed or non-compliance with regulatory requirements can lead to develop software not suitable for the application and therefore dangerous to the operator of the machine.

3.4 Intended use

⚠ Safety modules Gemnis series have been developed to carry out their functions in industrial applications that are in compliance with EN ISO 13849-1 or EN IEC 62061. Any other use is excluded.

⚠ These products are intended to be electrically connected to devices of the machinery and, according to the developed program, activate electrical outputs. The execution of these functions requires that the module has been properly installed and programmed, keeping in mind the instructions in this manual, the instructions manual of the module and all related components.

⚠ Safe state of Gemnis series modules is with safety outputs open (PNP safety outputs off or open contacts in case of relay output).

3.5 User's responsibility

The programmer operates in the configuration of the module by:

- defining the type of external components (the "sensors") connected to the module
- creating paths (the "connections") that the information gathered from sensors make to be processed by internal logic ("functional blocks")
- selecting the outputs of the module (the “outputs”) which will be activated.

⚠ The area of responsibility of Pizzato Elettrica in the programming process is only about the functions performed by individual procedures or functions performed by sensors, functional blocks and exits. It is entirely the responsibility of the user the choice, parameterization and interconnection of these procedures.

⚠ Knowledge of machinery by experienced operators can bring, in the risk analysis phase, to exclude certain types of faults. It is recalled that the liability for fault exclusion falls entirely on the machine's manufacturer.

2024 © Pizzato Elettrica S.r.l.

⚠ During the development of a program using Gemnis studio are possible multiple configuration options ("parameterization "), some of which have default values set by Pizzato Elettrica according to average use machinery manufacturers. Some examples, not exhaustive, of these values can be the maximum times of contemporaneity, contacts types, etc. These defaults values may be incorrect for certain types of machinery and it is the responsibility of the manufacturer of the machinery to check and if necessary change them.

⚠ Programs developed by Gemnis Studio can detect a significant number of failures of external devices connected to the module (electrical, mechanical and functional failures) but the opportunity and the responsibility to decide whether to detect them or not is delegated to the developer of the software because it is not possible to pre-determine how failures should be managed, as it is not possible to pre- determine fault exclusions or functional failure that exclusively result, case by case, from the analysis of each machinery.

⚠ Pizzato Elettrica releases the license for using Gemnis Studio or other related software for the configuration of safety modules which belong of the Gemnis series. The graphical interface of Gemnis Studio is mediated and controlled by other software (e.g. the "operating system" but not only) that govern the computer on which Gemnis Studio is installed. Sometimes, even without the user's knowledge, for example for those programs called "Virus", more software interact with the GUI and the memory of the computer on which you installed Gemnis Studio. It follows that Pizzato Elettrica cannot guarantee the correct computer graphic representation of the information in the module and cannot even prevent third party programs to alter the structure of the data of Gemnis Studio on the computer and then prevent them from providing incorrect information about programming that you are performing. For this reason, it is always necessary for the user to perform tests with the programmed modules to verify that their behaviour is exactly what it is expected, this test must be done before putting into service of machinery that uses them (see also 3.7).

3.6 Management of damaged or faulty products

⚠ If the goods have been visibly damaged during transportation or installation you mustn’t use them, not even for a test.

⚠ If after installing the module does not switch on or indicates an internal failure or has an unexpected behaviour it must be disconnected from the machine.

⚠ Repairs or modifications on the safety modules are prohibited. Removing or breaking the security seal invalidates the warranty on the product.

3.7 Validation and periodic tests

It is good practice and obligation according to norms, to make initial and periodic testing of safety devices on machinery. The fact that failure will not occur for a long period does not warrant that the safety circuit cannot fail. Think of the case of a circuit in which a push button for emergency stops has been wrongly wired such that the internal contact (normally closed) is not mechanically connected to the button in a correct way. Such a device may remain unused for years and make machinery look like as fully functional, even if there is a dangerous failure since the construction of machinery.

⚠ All programs developed by Gemnis Studio must be tested and validated on the machine before putting them in service. Testing and validation should also be repeated after each modification of the safety configuration of machinery both in case the change happening in the software that is loaded

 2024 © Pizzato Elettrica S.r.l.

intothe module whether it is done in the "hardware" part of the machine. Please note that many industrial components with the same function have variations in electrical parameters (contacts rebounds, response times) and safety parameters (PFHd, B10d).

⚠ It is necessary that all external devices connected to the safety module (buttons, sensors, barriers, etc.) are actuated or tested at least once a year (routine tests) in order to ensure that the probability ofdangerous failure, which is a function of time, can increase up to downgrade the safety circuit SIL or PL.

3.8 Directives and standards

The manufacturer who wishes to install a safety module series Gemnis to oversee all or some safety functions of machinery is responsible for verifying the conformity of all European directives applicable to machinery before its commissioning.

What follows is an exemplary and not limited list of applicable directives to a machinery:

Machinery directive 2006/42/EC

Low voltage Directive 2014/35/EU

EMC Directive 2014/30/EU

To apply the CE marking on machinery, the manufacturer may decide to use harmonized EN standards appropriate to the machinery. We remind some of the major standards used for safe design and risk assessment of machinery:

Standard	Content
EN ISO 13849-1/2	Safety of machinery — Safety-related parts of control systems
EN ISO 13850	Safety of machinery — Emergency stop function
EN ISO 12100	Safety of machinery — General principles for design
EN IEC 62061	Safety of machinery — Functional safety of safety-related control systems
EN 60204-1	Safety of machinery — Electrical equipment of machines
EN ISO 14119	Safety of machinery — Interlocking devices associated with guards
EN ISO 13851	Safety of machinery — Two-hand control devices
EN ISO 13857	Safety of machinery — Safety distances to prevent hazard zones being reached by upper and lower limbs
EN ISO 13855	Safety of machinery — Positioning of safeguards with respect to the approach speeds of parts of the human body

Please note that the manufacturer of the machinery who wants to certify the safety degree of a protected hazardous machinery, should make the analyses required by the safety standards on all different parts of the safety circuit including therefore also detecting devices, wirings, actuator devices, and not to limit the analysis to the characteristics of the safety module only.

2024 © Pizzato Elettrica S.r.l.

3.9 Limits of liability

This manual has been prepared and verified to the best of our knowledge. This does not imply that cannot exist inaccuracies or errors or that the text cannot be improved. Your comments are welcome and will be valued. In the site www.gemnis.com you can check if there is an updated version of this manual and tools are available to provide suggestions or reports of errors. The registration in the website also allows us to inform you in case of software versions with new features, improvements, bug resolutions or new versions of the hardware, are released.

Pizzato Elettrica provides this documentation only to customers which purchase modules of Gemnis series but this does not imply that Pizzato Elettrica has obligations to provide further documentation or support or new software tools in addition to the license of this version of Gemnis Studio.

Pizzato Elettrica reserves the right to modify and edit at any time, without notice and at its discretion, this manual and all data contained.

All the information contained in this document does not imply that Pizzato Elettrica assume responsibilities or liabilities other than as provided in the Sales Terms and Conditions indicated in the Pizzato Elettrica General Catalogue. Any further liability must be confirmed in writing by Pizzato Elettrica.

4 Gemnis line modules

4.1 Introduction

Project Gemnis modules are born with high flexibility skills, even regarding the hardware side.

These products are composed of a variety of electronic boards that are sold in different combinations but always contained in a single housing, and with a single product code.

This project, thanks to its modularity, permits flexibility also for the hardware level, in fact there's now the possibility to create ad hoc boards to customer specifications. Many companies in fact need to convert the dedicated hardware of their machine to hardware in accordance with new regulations. To perform this conversion a lot of resources are needed, profound knowledge of new regulations, important certification costs. If the customer does not find the correct electronic board among those already available can ask for an evaluation to Pizzato Elettrica having the certainty that the development (and related costs) does not cover the entire system but the single board in question.

4.2 General hardware characteristics

Gemnis series modules have a redundant and self-controlled structure. They are controlled by a pair of processors that run in parallel the Application Program and constantly monitor their functioning and system integrity.

Hardware Structure

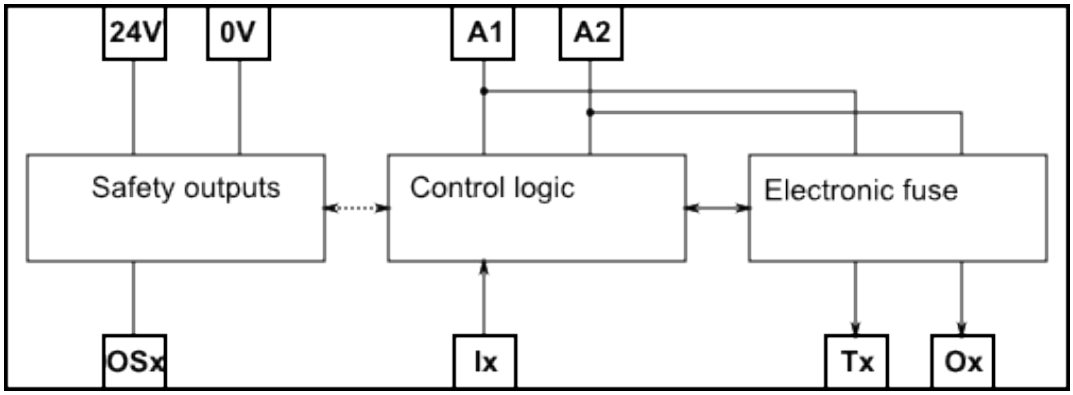
Each module comes in a single enclosure with the minimum width necessary to hold the boards that form the module. Enclosures ranges from 45 to 180 mm wide. The customer must therefore not worry about wiring different parts.

The layout of connection clamps on two rows on each side of the product allows the halving of space compared to similar products on the market. All the terminals of the modules are plug-in type and the rational arrangement of the terminals with the Test outputs and related inputs terminals allows a simplified wiring. The clamps may be requested in spring-type connection.

 2024 © Pizzato Elettrica S.r.l.

Power Supply

The structure of modules provides two power supply ports, dedicated respectively to the module itself (terminals A1-A2) and to safety outputs (terminals 0-24V). The two power supply circuits are completely decoupled and allow, if necessary, to manage different grounds between the two circuits. The main power supply provides energy to the control logics and, through an internal branch (see figure), to unsafe outputs ("O" type outputs) and Test signals ("T" type outputs). In particular, this branch is equipped with an electronic circuit overload detection, visible to control processors, function that allows the detection of short circuit to ground.



Clamps and connections

The clamps of the module are named according to their function. There are several different basic functions:

- Clamps “A” or “Supply”, dedicated to the power supply of the module logic. In some modules may exist more than one clamp A, for example when the module contains cards with decoupled power supplies. The odd numbered clamps (A1, A3, A5, etc..) are related to the positive supply, while the even numbered clamps (A2, A4, A6, etc..) are related to the respective mass.
- Clamps "I" or "Input" type, dedicated to the detection of input signals. A terminal type "I" in the module has a name such as "Ix" (e.g. "I11") where "x" uniquely identifies the clamp. The numbering of clamps is made according to the topology of the module, and then may not be consecutive.
- Clamp type "J" dedicated to the detection of input signals but decoupled (through photo couplers) with respect to the mass of the ground A2 of the module. For its ground clamp refer to the internal block diagram in the documentation of the module. A clamp "J" in the module has a name like "Jx" (eg "J51") where "x" uniquely identifies the clamp. The clamps numbering is done based on the topology of the module and therefore can’t be consecutive. Please pay attention that the test signals of the module, being generated with respect to the ground clamp A2, can’t be closed on the terminals "J" if you do not put in connection the respective ground with A2.
- Clamps “C” or “Current”. They are clamps specific to certain modules, dedicated to the detection of analog signals of type 4-20 mA.
- Clamps “F” or “Frequency”. They are clamps specific to certain modules, dedicated to the detection of signals from devices with digital out with frequency up to 4 KHz.
- Clamps "T" or "Test" type, i.e. the output control signals, which must be brought to the sensors (e.g. switch), and then come back in the module inputs. A “T” type clamp has a name like "Tx" (e.g. "T21") where "x" uniquely identifies the clamp. The numbering of the T-type clamp is made according to the topology of the module, and then may not always be consecutive. The clamps of type T are always present in pairs with consecutive odd and even index (e.g. "T21", "T22") for generating pulsed signal in phase opposition.
- Clamps "O" or "Output" type, the not safe outputs of the module. These clamps are identified by a

 2024 © Pizzato Elettrica S.r.l.

name like "Ox" (e.g. "O01") where "x" "uniquely identifies the clamp.

- Clamps "OS" or "Output Safe" type output, the safe outputs of the module. In the case of electronic outputs clamps are identified like "OSx" (e.g. "OS1") while in the case of relay outputs the clamps are in pairs and are called "x3-x4" (e.g. "13-14").

Programmable modules (modules which start with "CS MP" codes) are equipped with a USB port on the front. This port is used by the program "Gemnis Studio" to program the module and to detect its working state.

Status signaling

Each module has several green LED for signaling system status. The "PWR" led gives an indication of the presence of the main power supply (terminals A1-A2) of the module. The "Ixx" LEDs show the status signals on its input clamps. These two types of LEDs are not controlled by processors. Two LEDs (P1 and P2, both red/blue colors) are controlled directly by the two processors to highlight the status of the module and to provide information relating to identified faults. Some boards are equipped with dedicated LEDs. Please see the documentation.

Response time

The firmware contained in these modules has a cycle time of 10 ms. Execution of a safety function requires a minimum of two cycles, one for detection of inputs and the second for the execution of the Application Program (including the setting of outputs). Average execution time of a function is 20 ms. In general, the maximum response time of the system is always less than 30 ms. Some functions can also be performed in faster average times depending on the type of input signal and if using electronic safe outputs. Filtering on input signals can delay the average response time of the system.

Inputs

Cycle time equal to 10 ms is also the inferior temporal limit for detecting input signals. A signal, wired to an input, with a duration of less than 10 ms may not be detected.

The inputs are also filtered against the voltage micro-interruptions up to a maximum length of 0.5 ms.

4.3 Detection principles of external inputs and faults

The main types of fault that may occur in electrical devices external to the module are:

- Electrical failures: disconnecting wires, short-circuiting between wires, short-circuiting to ground, short circuit to power supply. An electrical fault inside the external devices led also to this type of failure.
- Functional failure: mechanical breaks or parts disconnection such that it retains the integrity of electrical circuit but not the integrity of the protective function. For example, the loosening of an NC electrical contact from an emergency stop mushroom push button.

To perform the safety functions and surveillance from faults listed above, in this family of modules are adopted the following general principles.

4.3.1 General principles for external fault detection

From a physical point of view, it is assumed that the presence of an electrical signal ("1") is a safe signal (necessary but not sufficient condition) and the lack of an electrical signal ("0") is not. This assumption prevents that a wiring disconnection or shorting a conductor to ground can be rated as a safe signal (these two events are the most common failure modes) while not excluding the case of a short circuit between two conductors or towards the power supply.

 2024 © Pizzato Elettrica S.r.l.

As a general principle it is assumed that it is always needed more than a single physical input (typically two) for the validation of a safety function, at least for a function having the maximum safety level. This highlights the functional failures, if not immediately at least on the first request, and by coherence analysis of the two signals even single failures towards the power supply.

From these two assumptions it follows that in general is always needed the presence of at least one electrical signal to activate a safety function. For example, this means that controlling a magnetic sensor at maximum safety level must be executed through the analysis of at least two electrical signals of which at least one is always present, i.e. the sensor contacts can be of type 2NC or 1NC + 1NO but not 2NO (with closed protection).

To evaluate electromechanical devices with electrical contacts in general it used the principle of creating "electrical loop" controlled by the module. The electrical contact is connected to a signal source created by the module (Test signal) and to an input clamp. The Test signal is sent into the device's contact and detected from the input stage. Evaluating the consistency between the Test signal and the value read, the system is able to detect the passage of a dynamic signal through the contact and therefore to exclude even short-circuiting persistent towards fixed tension. To discriminate the short-circuiting between two conductors it is used the principle of linking them to two different Test signals sources, typically in phase opposition, such that it is possible to identify how the signal sent differs from the read one. For this reason, the test signals clamps are always in pairs (odd clamp, even clamp) with signal in phase opposition.

To evaluate the electronic output devices in general it is not possible to create electrical loop. Most of the safety devices are equipped with dual electronic outputs with independent verification of outgoing signals (e.g. optical barriers in category 4 with OSSD type outputs) such that it is sufficient to verify the coherence of the input signals. Other devices have only single electronic outputs and, individually taken, they are not suitable to perform safety functions. However, the use of a variety of these devices, through temporal verification functions and sequential consistency, would permit their use in safety function. For example, the use of normal optical sensors (in the sense that they are not safety sensors) in muting functions.

4.3.1.1 About short-circuit between different inputs

Gemnis series modules adopt various strategies to detect failures. A possible fault is the external short circuit between any two input signals connected to any two Test signals. This failure is detected by the module through the analysis of Test signals generated and detected input signals. When an input signal is present but its Test signal is off we are in the presence of a short circuit to power supply or toward other Test signals. So, any short circuit between wires of a device connected to two inputs and two opposite Test signals is detected in a short time. On the other hand, in a module are connected many external devices and Test signals are often reused by them. A particularly unlucky case of failure occurs when two different devices that use the same test signals, have a short circuit between wirings after the devices themselves (between the device and the inputs). A dual-channel device is resistant to this fault type as failure would be detected on first use of one of the two devices. Instead, a single-channel device could be completely bypassed by this type of failure that basically creates an "electric OR" between the two devices. To work around this failure mode, it is recommended the use of different test signals for single-channel devices that have the wires that make the same route or physical separation of the wirings of such devices in order to exclude this fault.

Also, all Test patterns are repeated in pairs (odd index or even index) in phase opposition. So, all even-numbered Test signals and all odd-numbered Test signals behave the same way and it is not immediately possible to detect a short circuit between two inputs connected to test signals of the same type. For example, if two single-channel contacts are both linked to odd Test signals, a short circuit between the two inputs is not immediately detectable. To work around this type of failure Gemnis series modules introduce cyclical changes in all Test signals. Alterations occurs in sequence and the time to scan all Test signals is about 3 seconds, because it has been evaluated that breakdowns between different devices, or short circuits between the wires of different devices, are faults much less frequent of ones that may occur in individual devices.

 2024 © Pizzato Elettrica S.r.l.

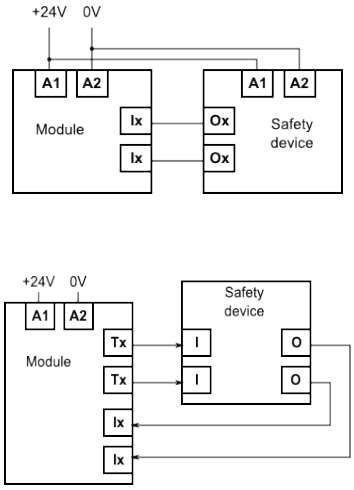
Finally, note how the concept of function and failure is related to the type of device in question. For example, consider the comparison between an emergency mushroom with 2 NC contacts and a safety mat with 4-wire technology. Both devices have the same electrical connections, but if the first normal operation provides for the opening of the contacts and a fault is for example a short circuit between the channels, in the second the situation is diametrically opposed because the short-circuit between channels is admitted in normal operation while the disconnection of one or both of the wires is a failure. To locate a fault, we understand therefore how it is necessary not only to evaluate the electrical nature signals but also having some information about the physical nature of the device. For this reason, it is necessary to clarify and differentiate two terms that are often used in this document: Safety Devices and sensors.

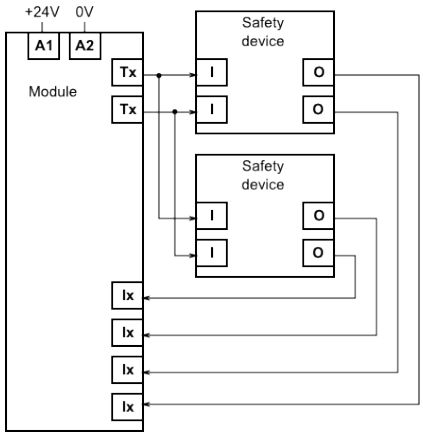
4.3.2 Safety devices

In this document the term "Safety Devices" means devices external to the module or, more generally, the systems of external devices that are assigned to the control of the dangerous parts of the machine and that are physically connected to the module clamps. Examples of safety devices are mushroom push button, safety switches, emergency barriers. Safety devices are also groups of devices, for example a pair of switches or two hand controls, which functionally serve to perform a single safety function for example safety control for one door. So, the term "Safety Device" is not always equivalent to a single physical object, but more generally identifies a system of devices designed to detect more signals that allow to carry out one safety function.

Also, a group of cascading electrically connected devices should be considered as one Safety Device. For example, a group of emergency push-buttons connected in series, or a group of RFID sensors connected in cascade are one safety device. The module alone is not able to differentiate the number of connected devices as the circuits are electrically indistinguishable.

Safety devices are typically connected to the clamps of the module, in general are connected to clamps of test signals and digital input clamps. A safety device may not use test signals (e.g. barriers) or use them in dedicated way (i.e. safety mats) or use them in common with other devices. A safety device uses at least one entrance and inputs of a safety device are not shared with other devices.





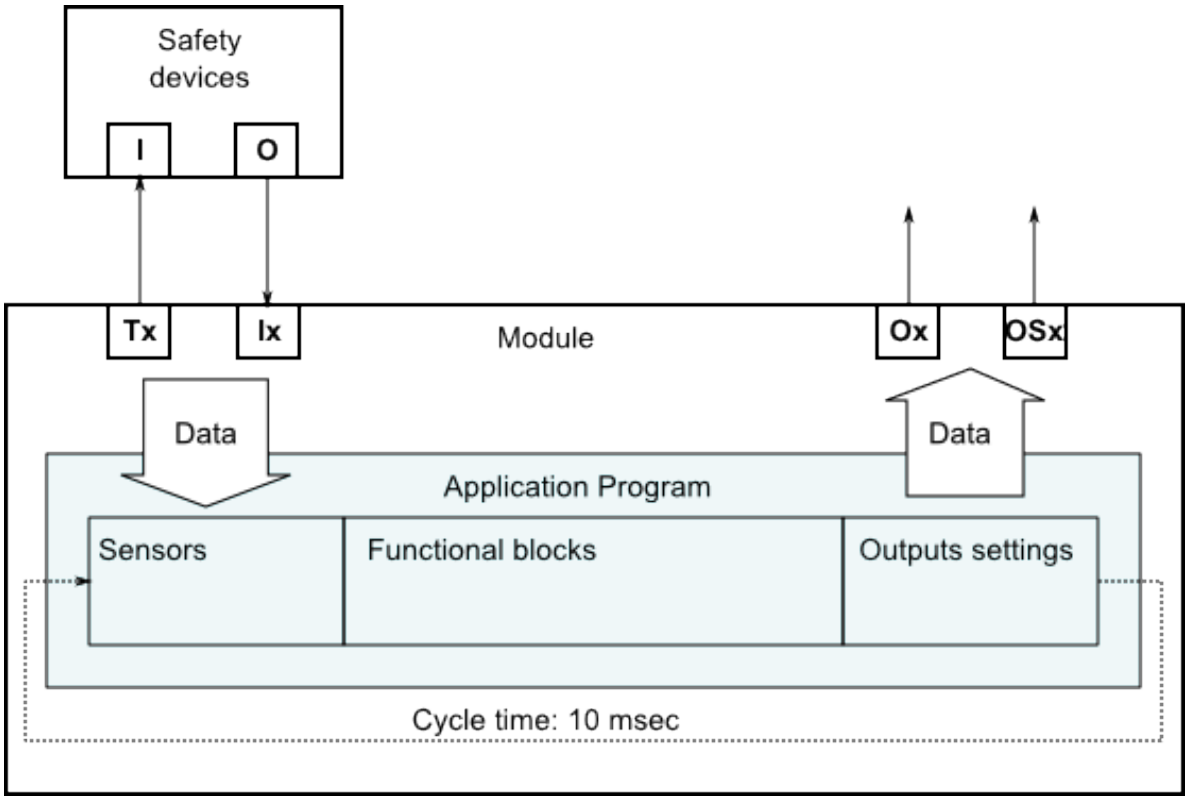
The module must process the received data from digital inputs in comparison with data of test signals to identify the status of operation of the device and any malfunction, but, to do this, requires further information relating to the physical nature of the device and it is for this reason that in the software was introduced the concept of Sensor.

 2024 © Pizzato Elettrica S.r.l.

4.3.3 Sensors

The sensors are part of the Application Program that serves to identify which type of Safety Device is physically connected to the clamps of the module. The sensors are chosen by the user within a default list depending on the nature of electrical safety device and are characterized by the user with all the additional information needed to fully define the behavior of the Safety Device.

The sensors are the first layer of software within a Gemnis module. At each execution cycle of Application Program, sensors analyze signals coming from outside the module to identify the status of Safety Devices and provide such data to the functional blocks that process them to end with the setting of outputs.

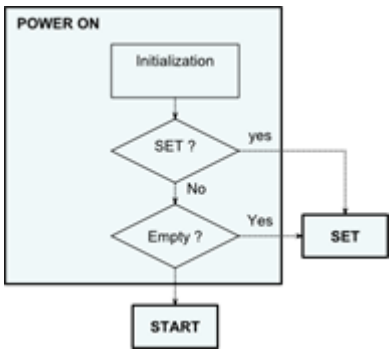


 2024 © Pizzato Elettrica S.r.l.

4.4 Modules state

The safety module, since its power on, can assume several States that characterize its functioning. The transition between States can be automatic or the result of internal processes or external signals.

4.4.1 POWER-ON State



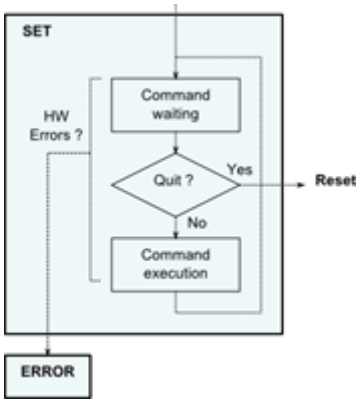
This is the State of on module at start-up or after a reset. In this state the module is in safe state, with all safety outputs open. During this phase it waits for approximately 1 second to check if in the USB port there is a demand for attention. If such request comes it enters in SET State. Otherwise, it verifies if the module is loaded with an Application Program. Where it is not programmed the module enter in SET state, otherwise START State. During the POWER-ON the modules does not detect any errors.

4.4.2 START State



This is the State that you have after a Power-On on a programmed module without request on the USB port. In this state the module is in safe state, with all safety outputs open. During this phase, the module performs an internal global verification both for software and hardware. If during the START phase the module detects an error it immediately enters into ERROR State, otherwise goes into RUN State.

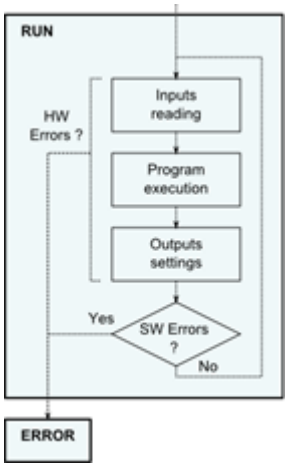
4.4.3 SET State



In this state the module is in safe state. Depending on the access level (password may be required) the module accepts query or set commands, such as loading a program or otherwise. It is possible to exit from SET State only through a physical shutdown of the module or by using a reset command. In this case the module starts by POWER-ON State (hot power-on).

 2024 © Pizzato Elettrica S.r.l.

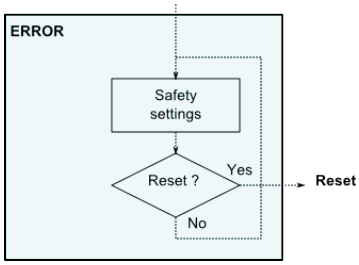
4.4.4 RUN State



In this state the module executes endlessly the Application Program, it reads the inputs and set outputs, reads commands from communication ports and, if enabled, writes responses.

Together with Application Program execution, module carries out on- line system test. If during this phase a hardware failure is detected or a software error is invoked the module enter in ERROR State.

4.4.5 ERROR State



In this state the module is in safe state. If detected failure is Hardware type, then you can reset module only through power supply. If the system goes into ERROR state following a request by the Application Program then the module can be reset through a reset command. If it receives the reset command the module start again form POWER-ON State.

Failures or errors that can bring a module in ERROR State are many and very different. In particular you can have software errors and hardware failures, which in turn can be detect internally or externally to the module. Among the failures detected outside the latter form we can distinguish between functional and electrical fault.

 2024 © Pizzato Elettrica S.r.l.

4.5 Module state display by P1 and P2 status LEDs

When powered up the PWR LED turns on and the module begins from POWER-ON state.

If the module goes into SET state, P1 and P2 LEDs, controlled by processors, flash alternately in blue. Otherwise, the module continues in START state and performs an internal self-diagnosis. In this phase the two LEDs (P1, P2) remain lit up with red light for about 1 second. If the test ends without internal anomalies, both LEDs will turn off and the module goes into RUN state and execute the Application Program. If the start-up tests are not passed, the module goes into ERROR state and LEDs remain lit up with red light to indicate the fault. Green LEDs for power and inputs of the module are not controlled by processors and begin immediately to indicate the status of its inputs / outputs.

When the module is in RUN mode and no faults are detected, the two LEDs (P1, P2) are off.

In the RUN mode, the module can detect the faults external to the module, for example due to short circuits, or invalid inputs conditions.

Some failures are managed directly from the hardware module such as in the case of overload or overheating. In this case the fault is highlighted by means of the blinking LED P1 and P2, with a specific flashing sequence that starts with o single red flash followed by N flashes of blue color, where N indicates the type of error detected (see table below).

Other faults are handled directly by the Application Program, typically faults on the Sensors. Depending on the type of fault detected, the Application Program may require the module to place itself in ERROR state, to highlight the anomaly, with a specific error code which can vary from 1 to 16. In this case the module shows the error code N via a sequence of N blue flashes by LED P1 and P2.

PWR	**LEDs P1 e P2**	**Possible fault cause**
Off ○ Green	Off ○	No power supply, incorrect connections, power wires cut, external fuses broken. Module fault.

 2024 © Pizzato Elettrica S.r.l.

4.6 List of products

Currently are defined the following products:

Code	Safety inputs Ix	Decoupled safety inputs Jx	Input 4-20 mA Cx	Frequency inputs Fx	Test outputs Tx	Safety outputs OSx	Auxiliary outputs Ox	Width (mm)
CS MP201M0	8	-	-	-	8	3NO	4	45
CS MP202M0	16	-	-	-	4	4 PNP	4	45
CS MP203M0	12	-	-	-	4	3NO + 1NO	4	45
CS MP204M0	12	-	-	-	4	3NO	4	45
CS MP205M0	4	4	-	4	4	4 PNP	4	45
CS MP206M0	8	-	-	-	4	4 PNP	12	45
CS MP207M0	4	-	2	-	4	4 PNP	4	45
CS MP208M0	16	-	-	-	4	8 PNP	-	45
CS MP301M0	24	-	-	-	8	3NO	4	67,5
CS MP302M0	24	-	-	-	12	4 PNP	4	67,5
CS MP303M0	32	-	-	-	4	4 PNP	4	67,5
CS MP304M0	28	-	-	-	4	3NO + 1NO	4	67,5
CS MP305M0	24	-	-	-	4	4 PNP	12	67,5
CS MP306M0	20	-	-	-	4	3NO + 1NO	12	67,5
CS MP307M0	8	4	2	4	4	4 PNP	4	67,5
CS MP308M0	24	-	-	-	4	8 PNP	8	67,5
CS MP309M0	32	-	-	-	4	8 PNP	-	67,5
CS MP310M0	8	8	-	8	4	4 PNP	4	67,5
CS MP311M0	20	-	2	-	4	4 PNP	4	67,5
CS MP312M0	16	4	-	4	8	8 PNP	0	67,5
CS MP401M0	40	-	-	-	4	4 PNP	12	90
CS MP402M0	32	-	-	-	12	8 PNP	8	90
CS MP403M0	40	-	-	-	4	8 PNP	8	90
CS MP406M0	32	-	-	-	4	4 PNP	20	90

All the products are equipped with USB port.

The list of products and boards may not be complete. Check the website www.gemnis.com to get updated information.

The customer who wishes to create a module with a new combination of Inputs/Outputs should contact the technical Office of Pizzato Elettrica.

 2024 © Pizzato Elettrica S.r.l.

4.6.1 USB Port

The USB port integrated in the module is used for programming and debugging the module by software Gemnis Studio. Once a module is programmed, you can use the USB port for communication with a PC installed on the machine and exchanging information relating to the state of the module. It is possible to communicate with the module by setting the USB port on your computer to emulate a serial port (COM) with the following parameters:

Speed: 9600 baud

parity: N

Data: 8 bits

Stop: 1 bit

Flow control: Xon/Xoff

For more information about the commands that the module accepts from USB see paragraph 6.

5 Gemnis Studio

The program Gemnis Studio is a graphical development environment for creating, simulating and debugging of programs suitable to be placed in the modules of Gemnis family. This software is released under license to the user that want to develop these modules after registering on the website www.gemnis.com. User registration is required to keep buyers informed on updates or notifications regarding the products used.

5.1 Prerequisites for performing of Gemnis Studio

The program Gemnis Studio is designed to operate on a personal computer with the following minimum requirements:

Operating system: Microsoft Windows 10, Microsoft Windows 11 in which must be installed framework .NET 4.6.1 or higher version and Microsoft Report Viewer. The framework .NET and Microsoft Report Viewer, if not already installed, are automatically installed during the installation of Gemnis Studio.

RAM memory: 256 MB

Disk space: 150 MB

USB connection: 1.0

Minimum display recommended resolution: 1280 x 800 pixels.

5.2 Software version

The installed version of the software is visible in the menu: ? → Info on Gemnis Studio

5.3 Introduction

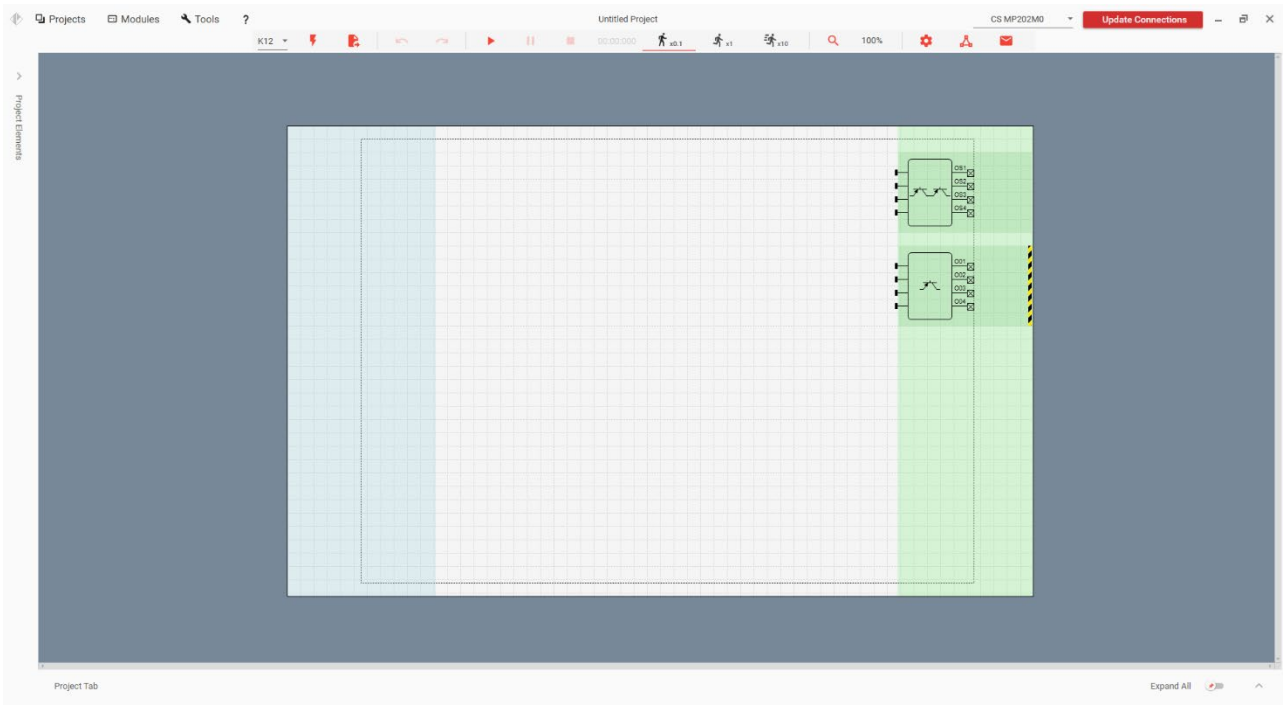
Gemnis Studio is a software developed to allow a user to program a module belonging to Gemnis series. This software has a graphical user interface to make visual, in a natural and intuitive way, the set of operations that the Application Program will run once loaded into the module. Gemnis Studio also allows

 2024 © Pizzato Elettrica S.r.l.

us to equip the configuration information with supporting information and useful notes for the complete understanding of the program. Finally, Gemnis Studio allows to perform debugging using the ability to simulate the operation of a module or detect and graphically represent the status of a real running device.

5.3.1 Main menu

Once started the program Gemnis Studio work environment is as follows:



The software at start-up automatically creates a new untitled project, where it is possible to start the design immediately.

The first menu bar contains four drop-down menus that are:

Projects: the function of Gemnis Studio that allows the creation and management of program configuration for the modules.

Modules: presents a list of Gemnis series modules currently being developed and their main features.

Tools: allows you to set some of the general characteristics of the Gemnis Studio software or the execution of general operations.

?: provides online help for the program and for additional data of Gemnis Studio and the access to the remote support function.

5.4 Tools Menu

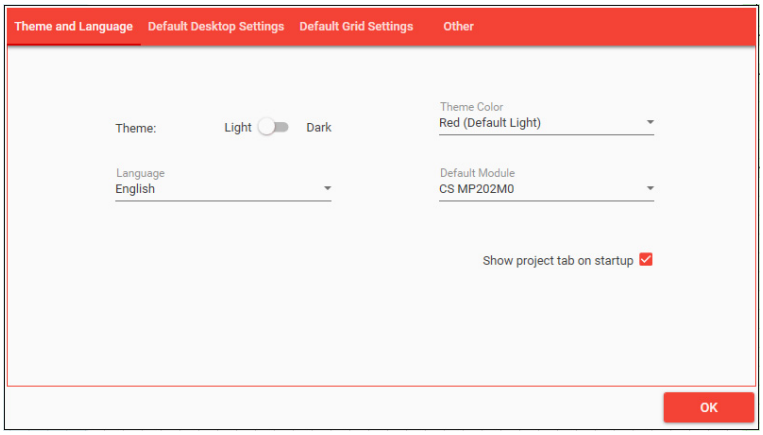
Allows you to set some of the General characteristics of the Gemnis Studio software or performing general tasks

5.4.1 General options

Selecting Tools, the first entry in the drop-down menu Option allows the choice of some general options for the program:

 2024 © Pizzato Elettrica S.r.l.

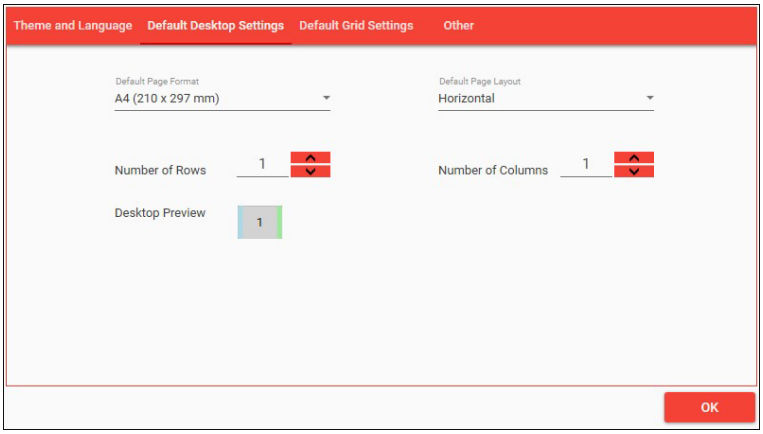
5.4.1.1 Theme and Language



Gemnis Studio has two themes available, Light and Dark, and for each one it is possible to customize the color through the drop-down list Theme Color.

Gemnis studio has been developed to be able to work with different languages, the selection of the language is possible through the drop-down list Language. Inside the tab Theme and Language, it is also possible to select the default module for new projects, in this way the module will be set automatically at the opening of a new project. In the end it is possible to check a box to have the project tab visible at the start-up of the software.

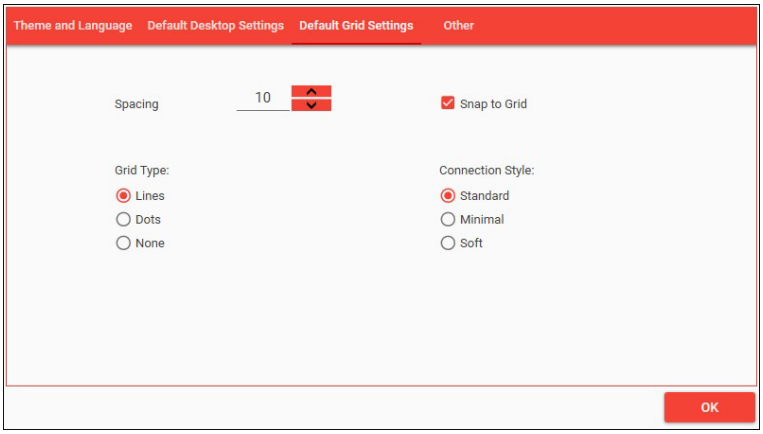
5.4.1.2 Default Desktop Settings



This section includes the working area settings. Gemnis Studio has been developed in order to have the working area which is always a printable format. It is possible to select the page format, the page orientation and how many sheets (rows and/or columns) the working area is composed by.

Inside the tab it is present a preview of the working area (Desktop Preview).

5.4.1.3 Default Grid Settings

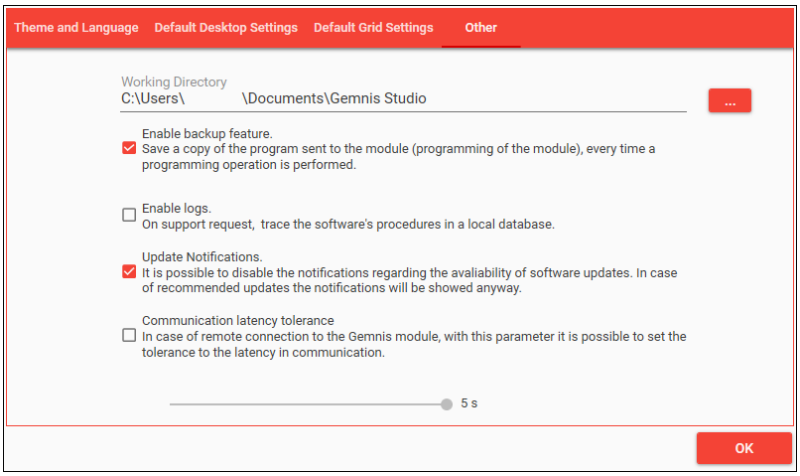


This section allows to define the general settings of the working area: the grid style (spacing and type) and the connection style. With the check box Snap to Grid selected every object inserted in the working area will be automatically placed at the closest grid coordinate.

These settings involve all the new projects. The saved projects will not be modified by these settings, and they can be modified in the project desktop settings (see 5.7.1).

 2024 © Pizzato Elettrica S.r.l.

5.4.1.4 Other



The section Other allows to set the Gemnis Studio default working folder. In this directory there will be saved all the Gemnis Studio projects. As default this folder is inside “Documents” and it is called “Gemnis Studio”.

By ticking Enable backup feature, it is possible to set Gemnis Studio to make an automatic save whenever this project is loaded in the module. The project is saved in a file with the same file-name of the project followed by the date of compiling within the sub directory "\Backup" of Gemnis Studio folder. If you need to open a project from a Backup file, please note that you must first rename the file with the name of the project, removing the part of the date.

In case of contact with the Technical Support it might be useful to track software’s activities in a log file; if the technician requires it, this functionality can be enabled through the check box Enable logs.

Update Notifications, verifies the presence in the network of a new minor update and inform the user through a window that will explain how to obtain it. This type of notifications will always show up for major updates.

The Communication latency tolerance allows to manage any latencies in the communication between Gemnis Studio and the security module in cases where the module is connected to the computer via a gateway for remote connection.

5.4.2 Extracting a program from a module

Through the Tools → Import program from the module it is possible to read from a module already programmed the Application Program (see Application program and digital signature), and save it to a file. In case you lost the source files of the project, in this way you can make Backup copies or load to other modules an identical program. The ability to extract data from the module depends also on knowledge of the Password, if present, registered for the protection of the Application Program.

After pressing the Start Import button, the software scans the COM ports and, if a module is connected, saves the program with the same name as the project inside it.

> ⓘ The extracted file is saved with the “.pzzke” extension which is the same as the user program. Therefore, it is recommended to be careful not to overwrite the source file.

Please note that the file extracted from a module doesn’t contain the information necessary for the reconstruction of graphical programming user interface. Consequently, this file can only be used as a backup copy or for loading the identical program in other modules.

 2024 © Pizzato Elettrica S.r.l.

5.4.3 Programming by rows of one or more modules

Through the tab Tools → Sequential Modules Programming you can program one or more modules directly from a Project File (see Projects) without going through the graphical user interface of Gemnis Studio. This option is useful for programming multiple modules directly in the production plants, without necessarily access users to graphically design source files.

5.4.4 Generating an extracted file

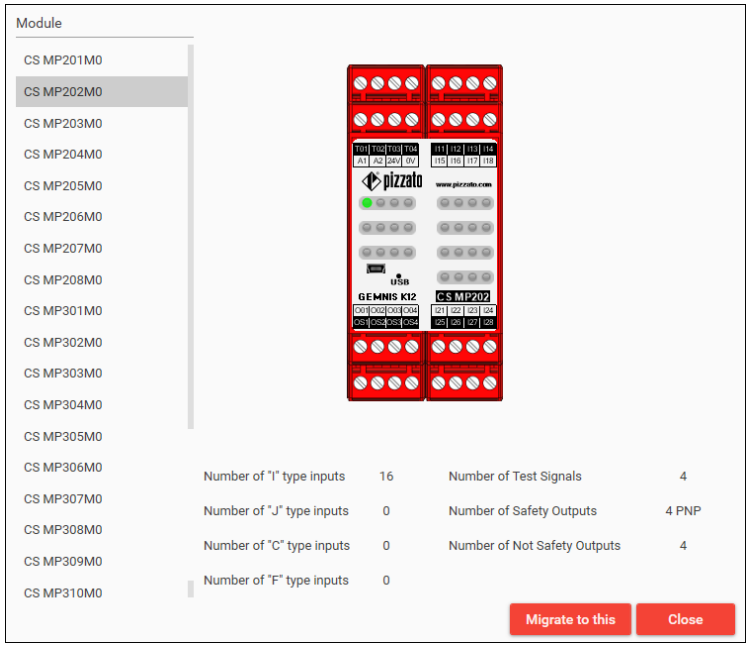
Through the tab Tools → Generate extract file you can save the Application Program (see Application Program and digital signature) of the project you are working on. This file does not contain the information necessary to reconstruct the graphical programming interface, consequently this file can only be used as a backup copy or to program other modules in an identical way.

> ⓘ The extracted file is saved with the ".pzzke" extension which is the same as the user program. Therefore, it is recommended to be careful not to overwrite the source file.

5.5 Modules Menu

From menu item Modules → View Modules you can view the list of Gemnis series modules currently developed and their hardware features, such as the number of inputs, outputs, etc.

If available, by clicking on the Migrate to this button, Gemnis Studio will ask the user to choose whether or not to migrate the project to the new module, or start a new project.



Visit our website www.gemnis.com to get the file with the updated list of modules. To update the list of modules in Gemnis Studio, select the menu item Modules → Update Modules List and select the file you downloaded.

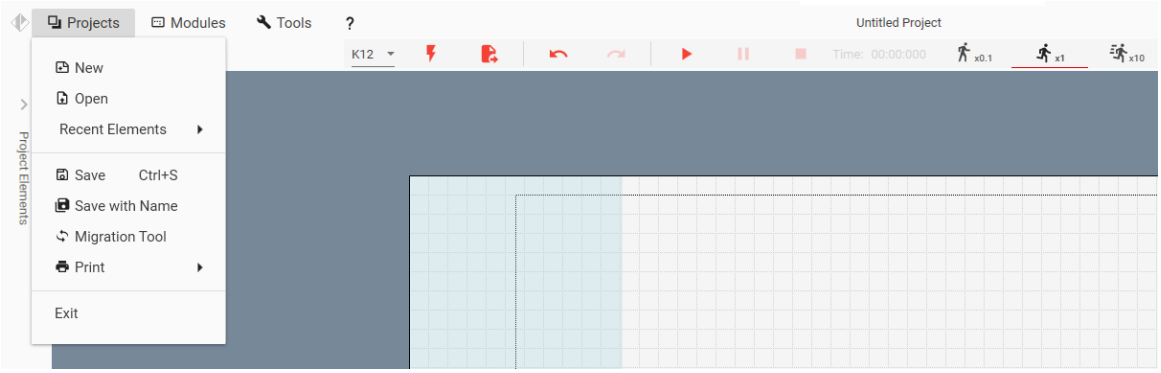
Clicking on the Migrate to this button it will initialize the procedure for migrating the project to the selected module (see Migration of programs).

 2024 © Pizzato Elettrica S.r.l.

5.6 Projects Menu

All the information you need to configure a module and to describe the activities is named "Project". Using Gemnis Studio the user can build an Assembly of textual and graphics information that allow you to describe and comment that will be performed when the program is installed on a Gemnis module. This collection of information is saved in a single file with the name chosen for this project and the extension "PZZKE". A file of type "PZZKE" is a project file.

At the start-up of Gemnis Studio, the software windows show a new untitled project based on the module CS MP••••• which has been set in the options window (Tools → Options). It is always possible to create a new project by selecting the menu item Projects → New



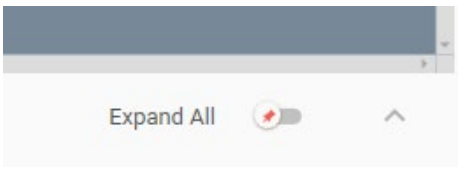
It is possible to change the module on which the program will be based through the drop-down list no the top right of the project tool bar (see 5.7.1).

The selection of the module is an important moment for the creation of a project, because resources represented within the project will be only the ones of the chosen module. The modules of the project Gemnis are not expandable, so if you need a different module for the realization of the project, it would be necessary to transfer what you created in a new project that is associated with the new module. Since different modules may have different names and resources, transfer may not be immediate. For this reason, before creating a new project, it is always recommended to perform an analysis of the resources necessary for its realization.

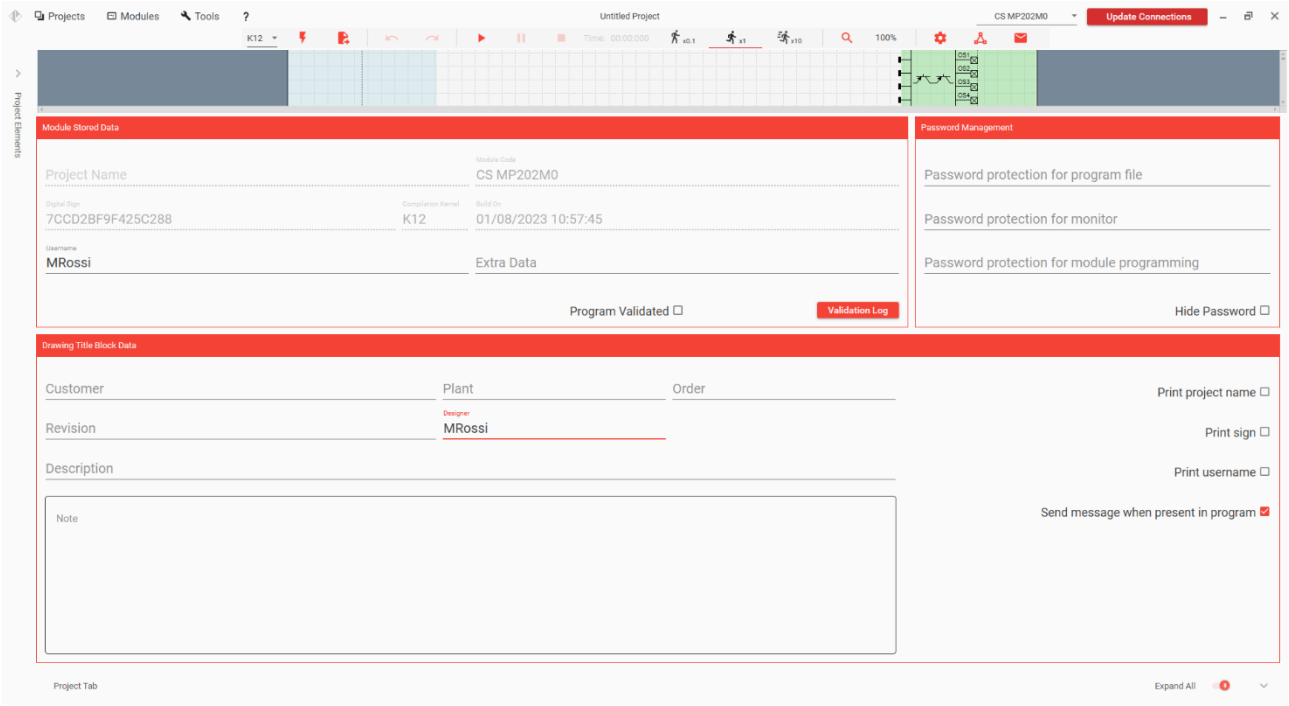
Gemnis Studio is a single project environment; however, it is possible to open multiple instances of the software.

5.6.1 Project Tab

The Project Tab is placed in the lower part of the software's window, it can be extended or hidden by clicking the symbol on the bar in the lower part of the window.



 2024 © Pizzato Elettrica S.r.l.



The "Project " tab contains general data relating to the project. The data inside the box "Module Stored Data" are part of the application program (see later) and that will be stored in the module. Other data are part of the project and will not be stored in the module but will be stored in the project file.

Some of the data stored in the module are represented on a grey background to indicate that they are not editable by the user. As you can see in the picture the "Project name", the "Digital Sign", the “Compilation Kernel”, the "Module Code" and the "Build On" data are not editable by the user.

The check "Program validated" and the button “Validation Log” regard the validation cycle of the project. This aspect will be discussed further in section 5.7.1 in "Analysis and transmission of the program."

"Username" (16 characters) and "Extra Data" (24 characters) are, on the contrary, can be editable. The "Username" field is pre-filled with the user name of the user of operating system.

The "Extra Data" field is needed for the user to enter data he wants (design, code, Reviews, ...) inside the module. Once programmed the module data can be checked by Gemnis Studio (see Monitor) or through external programs (see System commands).

All other fields are stored only in the project and they are freely editable.

The fields that appear in the "Title block data" are provided in print (see Prints).

The check boxes "Show project name", "Show signature", "Show user name" mean that the data stored in the module are also included in the title blocks of prints.

In "other information", the field "Description" is used to give a brief description in the project that will be used during the research of PZZKE files from the menu Projects → Open.

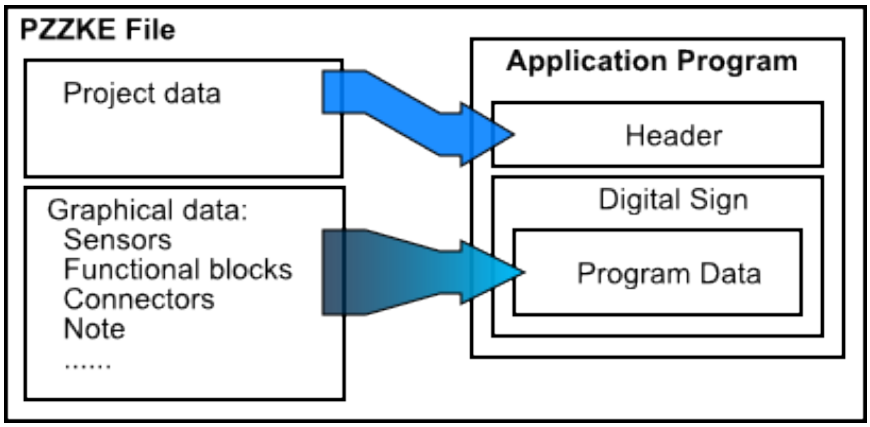
The "notes" field allows you to write notes in extended way.

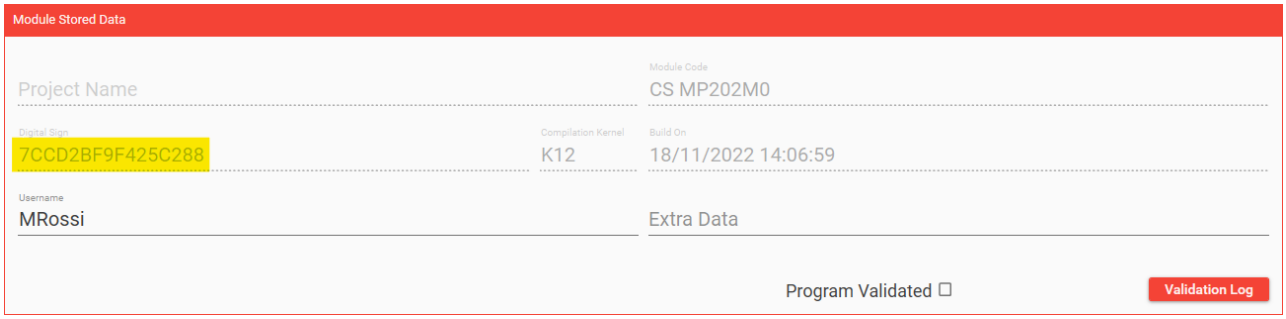
5.6.2 Application program and digital signature

The project file contains all the information of the project including those one not necessary for the operation of the module, such as the graphics and comments. This collection of information is excessive for the storage capacity of the module. For this reason, from all the information in the project file the module programming software will extract only those necessary for the proper functioning of the module and only

this information will be transferred to the module. This collection of information is called "Application Program".

The collection of data, necessary for the configuration of the module, is subjected to a mathematical calculation that extracts an identifier code called "digital signature", which is a string of 16 characters that uniquely represents the Application Program.





The digital signature is computed from all and only the information necessary to the proper operation of the module. This information includes the Kernel generation (10, 11 or 12), then the same project built for different Kernel generations will present different digital signatures. For example, a project generated and built with Gemnis Studio 11, once opened with Gemnis Studio 12 and re-built for Kernel 12 (without any program modification) will present a different digital signature than the first one.

For calculation of the digital signature are then excluded not only the information data present in the file PZZKE, but also some data of the headline that exist in the application program, but don’t contribute to the operation of the module. Are evaluated for calculation of digital signature the name of project, the build date, username, and the Extra data. The possibility to change these data without changing the file's digital signature allows the insertion of some information user variables (such as a unique serial number or other) in the configuration file, data that can then be adapted at any time by the module once programmed.

The digital signature of a project is recalculated at each compilation and always before creating an application program ready to be transferred in a modulo.

The project file must be protected and saved properly. In particular, it is recommended to run at regular intervals backup files of type PZZKE that, if deleted, can no longer be recovered. The application program, on the contrary, can be read again by a programmed module (only if you have permission, see next section), but these data do not allow a complete reconstruction project files with information and graphics. Such data may be used solely for Backup or recovery of other identical modules (see 5.4.3).

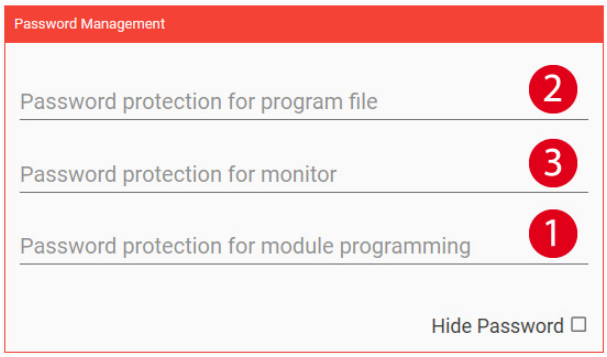
 2024 © Pizzato Elettrica S.r.l.

5.6.3 Password

Gemnis series modules are designed to provide a number of services and information through existing connections. In particular, modules with USB port can be programmed, providing information about the program loaded and its executing state, communicate internal variables values, be restarted and provide debugging information. The ability to perform all these operations is very useful in realization or in testing phases of a project but, once the project is completed and is loaded on the modules that are installed in the machinery, the programmer may wish to limit the possibility of alteration or reading of data in module for other people.

In other cases, the programmer who developed the project wants to give the project file to other people in the company only programming new modules, without, however, providing access to the source code or by limiting access to the only chance to see the monitor of the state of functioning of the module.

To meet these needs, it is possible protect by password the ability of interaction of a module and/or the ability to edit the project file.



Protection Password for the module (1)

A module can be programmed either with or without a password. In any case it is always guaranteed a minimum level of interaction or "access level" (see note) that allows the reading of the nominal data of the project, the State of the module (see 4.4) and the current Access Level.

> ⓘ In modules Gemnis series, is defined by the term "Access Level" the degree of interactivity in the form through the USB port. Access levels are in total 4, numbered from 0 to 3 in ascending order and this implies that if the module is placed at a certain level of access, all lower levels will be available too. For example, to access level 3 will be executable all the functions available access levels 0,1 and 2. Are currently used only the basic (level 0) and maximum (level 3) access levels.

Module not protected by password

When a module is programmed with an Application Program not equipped with password, the Access Level of the module (programmed) is the maximum level allowed, the level 3. This access level allows writing, reading or the cancellation of the application program. In this level you can also restart the software while you are running the application program.

When a module is not programmed it sets itself to level 3. This is also the condition of a module just purchased.

Module protected by Password

When a module is protected with password has Access level equal to 0. At this level you can invoke only a few basic commands and the command that allows you to change the level of access. Invoking this

 2024 © Pizzato Elettrica S.r.l.

command, the password that is compared to the in-memory module will be requested. If the password is identified in module the access is set to Level 3. If you do not re-program the module, the next reboot is set to access level 0.

At Access Level 0 you can read the value of the internal variables of the program (but not the underlying code) and therefore, also through specific external programs, monitor the operation of the module. This level of access is required to Gemnis Studio to graphically represent the State of the program at run-time. Setting the protection Password module takes place in the main form of the project.

When you create a new project there are no password and in this condition the form, once programmed, always access at Level 3. This condition is normally used in the development phase of a project so that you can quickly reconfigure a module without having to provide the password every programming.

When a project is completed and you want to protect module then you need to enter the password you want. The password may use any non-ASCII character (from 0x20 to 0x7E) and have a maximum length of 16 characters.

When the module is programmed and restarted, the module identifies the new access level and no longer provides accessibility to information and commands of a certain level if before is not provided your password to access this level.

Password protection of the project file (2)

The project file can be protected with a password to prevent modification of the source code (the graphics) by other users. Setting the password protection of the project, the PZZKE file can't be edited but it may be used to send the data to a module through the Sequential Programming Procedure as already explained in the paragraph Programming by rows of one or more modules.

For example, consider the case in which a programmer develops a project that will be installed in the modules by another person of the staff, typically directly on the production line. If the programmer does not want other people to edit the project, can provide such person a copy of the installation file Gemnis Studio and the project file (protected by a password).

Once set, this password will be required each time you open the file.

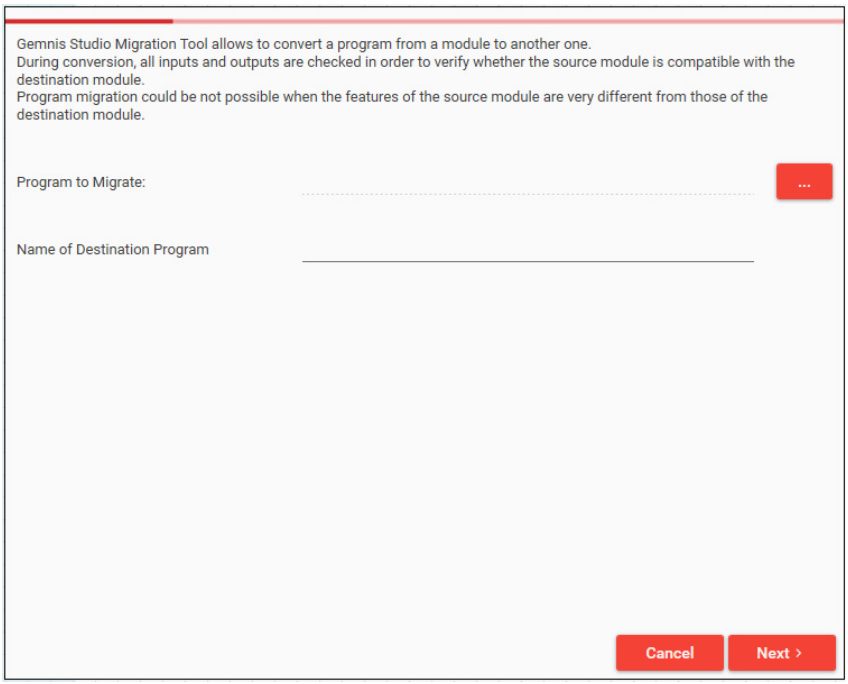
Password protection for the monitor mode (3)

In case you want the project file not to be changed and even not be displayed in Gemnis Studio monitor, you can set a new password. It will be requested every time you try to display the state of the module in the monitor.

 2024 © Pizzato Elettrica S.r.l.

5.6.4 Migration of programs

From the Projects menu it is possible to select the Migration Tool entry, in order to convert a program that had been created for a specific module and adapt it to a module other than the original one.



In order to migrate a program, it is necessary to:

1. select the file containing the program to be migrated;
2. enter a file name for the destination program;
3. select the destination module from the list of available modules.

During the conversion, Gemnis Studio performs a check on the inputs and outputs, in order to check the compatibility of the program between the source module and the destination module.

Program migration between modules may not be possible when the features of the source module are very different from those of the destination module.

Any incorrect or missing connections in the destination module are automatically removed from the application program.

> ⚠ Even if all the connections between the source module and destination module are correct, it is always recommended to check, analyze and correct the result of the compiled migrated program manually, in order to guarantee the correct execution of the application program.

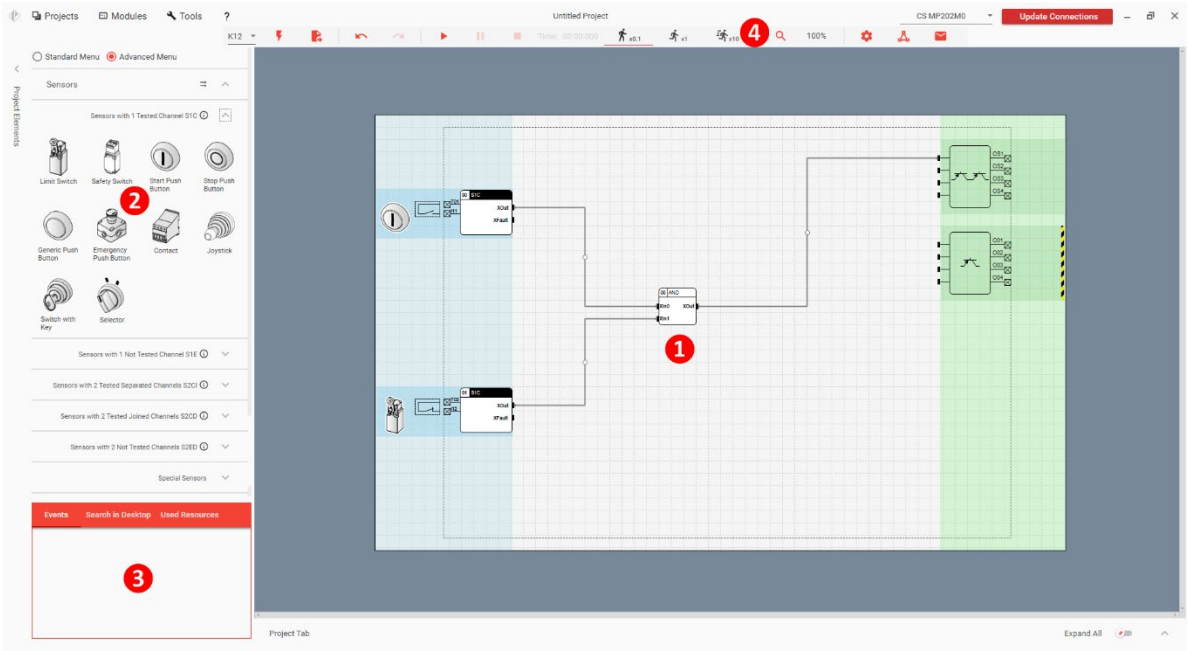
 2024 © Pizzato Elettrica S.r.l.

5.7 User Program

Once you have created a project you can start immediately creating the application program by accessing the "User Program" tab.

The work environment is so divided:

1. The Desktop that is the main workspace.
2. Panels with sensors, functional blocks, objects
3. Search in desktop, The Log of events and resources area
4. Bar with buttons for the compilation of the program loaded into the module, the simulation program, the ability to set some options for the desktop.



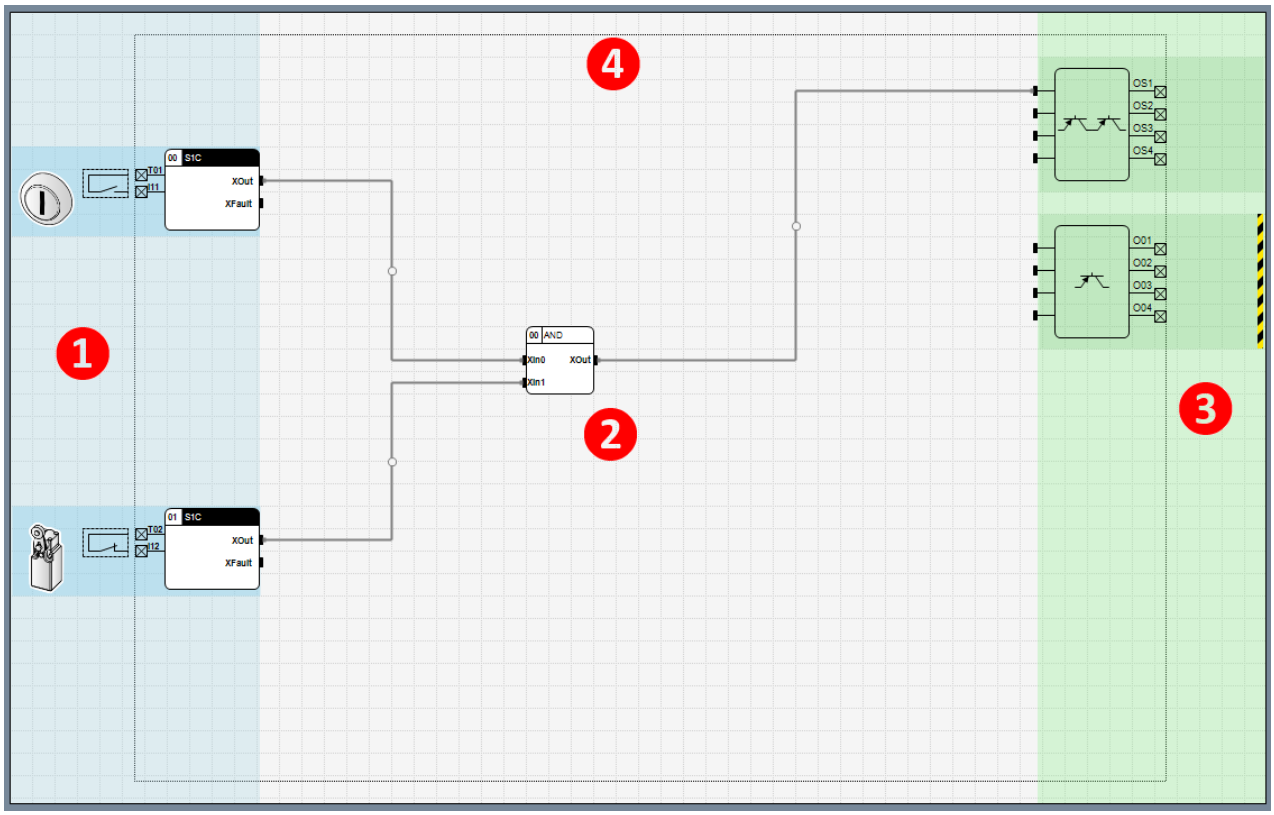
5.7.1 Desktop

Studio Gemnis was created with the aim of making the most immediate and intuitive the use of a Gemnis series module. With this target and reminding the typical application of these products, we have chosen to create a working environment, the Desktop, where the user has, as far as possible, all the information he needs to "see" and not "imagine" how works the project he is developing. For this reason, we tried to focus our attention to the graphical representations of the objects, the physical characteristics of the module used, immediate interaction, through simulation, with the program created.

The Desktop is the main workspace of the user, the area where is defined, using the graphical user interface of the program, the flow and processes that must be applied to the data detected by the module. The desktop is divided into three parts:

1. the sensor area
2. the functional blocks area
3. the outputs zone

 2024 © Pizzato Elettrica S.r.l.



In the Sensors area (1) the user will indicate which type of external devices are connected to the terminals of the module and all the parameters needed to define them.

In the area of Outputs (3) are present all output devices present in the chosen module (relays, transistors etc.).

In the area of functional blocks (2) you will insert all necessary logical functions to develop the flow of data from sensors and will make connections that transfer data between objects on the Desktop and on the outputs.

In the Desktop there is a dashed box (4) that represents the area “occupied by the module” that is, from clamps to the code, everything is enclosed in a physical module. The area outside of this box is instead occupied by images of physical devices external to the module (switches, buttons, etc.) by their internal structure and their expected eventual description.

The selection panels and the Event Log

On the left side of the desktop, there are some panels for the selection of the objects to include in the desktop and for information.

There are two different modes of visualization of the panels:

1. Standard Menu
2. Advanced Menu

In the Standard Menu the panels keep the traditional arrangement already present in the previous Gemnis Studio versions. This menu arrangement is useful for those who use Gemnis Studio for some time and like to maintain the same visualization style.

The Advanced Menu presents a new arrangement of the selection panels with four sliding panels, selectable by clicking on the relative bar:

- **Sensors**: The list of all objects that can be used in area 1 (see 5.7.2), grouped in six further panels for each electrical type.

 2024 © Pizzato Elettrica S.r.l.

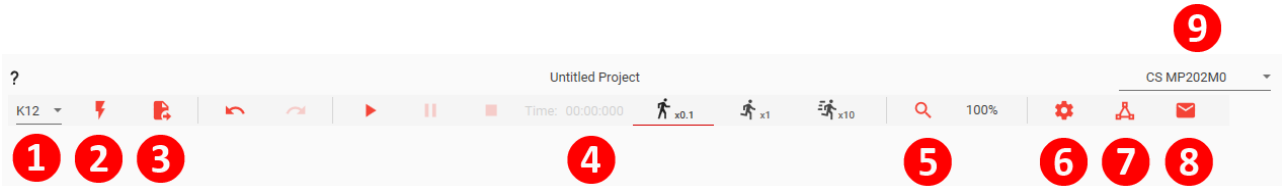
- **Basic functions**: The list of all objects that perform basic functions of safety and that can be used in area 2 (see 0). The basic functions include the logic gates and other functional blocks like START, DELAY and PULSE.
- **Advanced functions**: The list of all objects that perform advanced functions of safety and that can be used in area 2 (see 0). The advanced functions include all the rest of functional blocks, that are not included in the basic functions.
- **Objects**: The list of all objects that don't perform specific functions of safety but which are useful for completing or understanding of the functions performed out and that can be placed in the area 2 (see 5.7.6).

Below the sliding panels of selection there is an area with two tabs:

- **Desktop Search**: You can search for an object in the name or description within the desktop. The search is activated by pressing Ctrl + F in ' User program' which will place the focus directly on the search box automatically pre-selecting any existing text. The search is performed after pressing the 'Enter' key. Will present the results in the grid below, showing the type of the object found and the value of the property for extended that contains the value sought. By clicking on a result will be automatically selected the object and the desktop will be placed so as to be centered on the object. The identification of this is highlighted with a colored rectangle that gradually disappears. Any change to the desktop will reset the previous research
- **Events**: Show visually, with colorful symbols, a list of recent events of the program, such as the successful compilation of the project, the start of simulation phase, errors and others useful information.
- **Used resources**: After the compilation of a project, this area shows how many resources are used by the project. The area indicates respectively how many memories of type X, B e W, how many memories dedicated to the sensors and how much of the total storage of the module has been used, all of these data as a percentage compared to the maximum available.

5.7.1.1 Toolbar of the desktop

At the top of the desktop there is a toolbar that allows you to interact with the data in the desktop.



1- Kernel version

Allows you to choose for which kernel to compile the project. If you are programming safety modules with Kernel 10, you must select the value "K10", for modules with Kernel 11 you must select "K11", the default choice is instead the most recent "K12". A program compiled for a specific Kernel can't be downloaded to a module with a different Kernel.

2- Compile the program

It allows you to perform a compilation of the program on which you are working. The result of the compilation is visible in the logs that are displayed in the area previously described in section 5.7. The compilation of the program can give positive or negative depending on whether or not the program contains errors. Incorrect compilation does not allow the sending of the program to the module until the user does not correct the errors reported.

 2024 © Pizzato Elettrica S.r.l.

3- Analysis and transmission of the program

Performs compilation and displays the formal evaluation of the project. If there is a module connected, the user can send the program to the module.

4- Play - Pause - Reset - Time – x0,1 - x1 - x10

These buttons let you start, pause or interrupt the project simulation (see Simulation).

5- Zoom & Pan

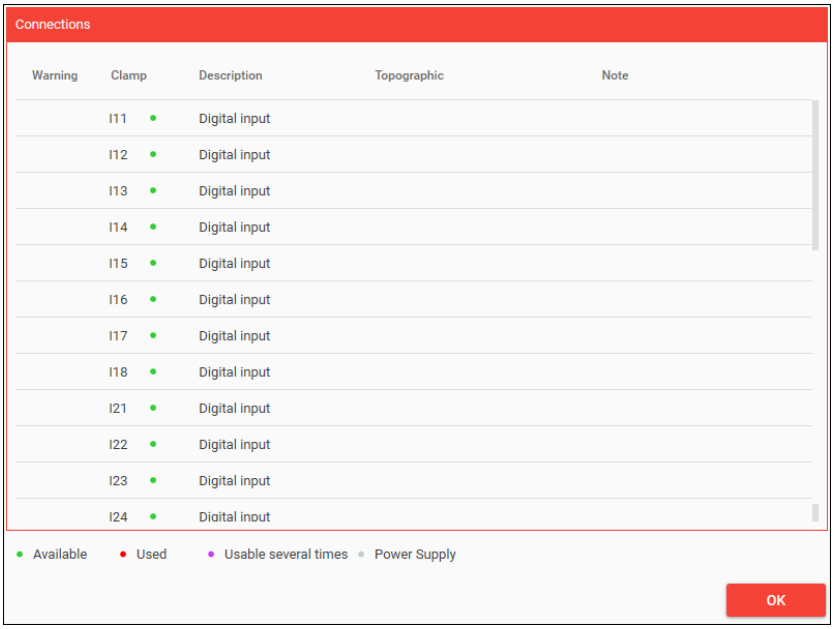
The entire desktop is zoomable through the combined use of the CTRL key and scroll wheel on the mouse, from 20% to 300% of normal size. Zooming is centered in the arrow of the mouse on the desktop. You can also manually set the zoom percentage you want, within the limits specified, using the zoom box. The Desktop is also movable (Pan) through the combined use of the Ctrl key (or the spacebar) and moving the mouse while holding down the left mouse button. At any time, you can return to normal size (100%) and to original location of the desktop by clicking on the icon Zoom. The Pan & Zoom functions are also available during the simulation of the project and in the Monitor tab.

6- Desktop settings

This button activates a form with the general graphics settings related to the project. You can indicate the type (size) and the orientation of the sheet that is under the Desktop and then that will be used during printing. By changing the type and orientation of the paper will therefore also changes the area of the Desktop and will be prompted to choose whether or not you want to recalculate the objects position. You can also change other graphics parameters of the Desktop.

7- Connections Tab

In the Connections tab are listed all physical terminals of the module and you can link to each of them one description and then print a report for technicians who have to build the electrical connections of the machinery.



 2024 © Pizzato Elettrica S.r.l.

Next to the clamp name there is a colored dot that indicates the status of the connection. Below there is a description of the various colors.

Color	Description	Applied to
●	The green dot indicates that the connection is free, and it isn't used in the program yet.	Digital inputs (Ix) Digital inputs, decoupled (Jx) Frequency inputs (Fxx) Analog Inputs (Cx) Signaling outputs (Ox) Safety outputs (OSx) Safety outputs (13-14, ecc.)
●	The red dot indicates that the connection is already used in the program.	Digital inputs (Ix) Digital inputs, decoupled (Jx) Frequency inputs (Fxx) Analog Inputs (Cx) Signaling outputs (Ox) Safety outputs (OSx) Safety outputs (13-14, ecc.)
●	The purple dot indicates that the connection is usable several times	Test signals (Txx)
●	The grey dots are dedicated to the power supply clamps	Power supply (A1, A2, 24V, 0V)

The "warning" field is used just as a reminder. In this field the symbol appears to indicate terminal attention that were recently modified in the project, e.g. after the insertion of a sensor using that terminal. These symbols disappear after editing text in row or after a saving the project.

"Topographic" field is connected with descriptions on the Desktop. For instance, changing the description of clamp "O01" on your Desktop, you will result that same description will appear in the field in this tab and vice versa.

The "Notes" field is freely editable.

The printing of this report is available from the menu "Prints" (see 5.9).

8- Messages Tab

In Messages tab are stored the messages eventually transmitted by the functional block MESSAGE (see 5.7.3.20).

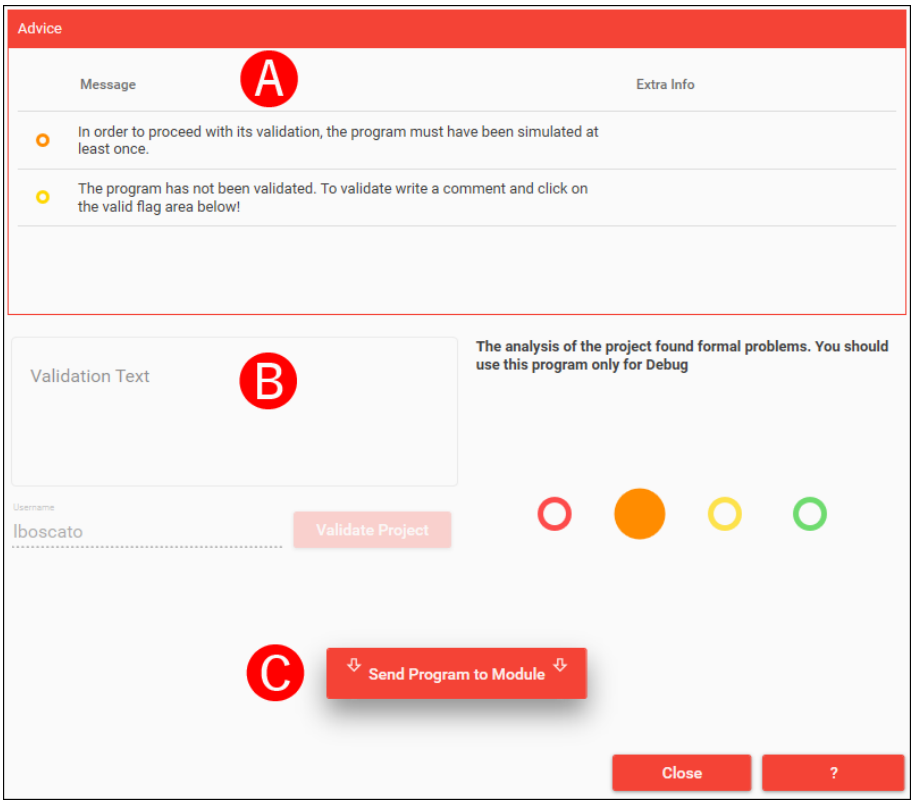
9- Module selection

It is possible to select the safety module of the CS MP series to be programmed. In case it is necessary to change the module after having already designed a part of the program it will be prompted to choose whether or not you want to migrate the project on the new module, or to start a new project.

 2024 © Pizzato Elettrica S.r.l.

5.7.1.2 Analysis and validation of the project

The window for analysis and sending of the program is shown in the following figure.



A. Formal evaluation: this feature is intended to help the programmer to avoid omissions or inaccuracies. It is basically a formal judgment on the whole project. This evaluation does not prevent the programming of the module, even if the error-free program is not considered formally correct, as may exist extremely different cases which cannot be evaluated by the program. For example, the program can be only a test to be tested on the desk of the designer, or some reports may not be reflected in real life due to technological choices or fault exclusions that can be made only by those who know the machine on which the module will be installed. In user help is a visual representation of the results of evaluation described above. The colors red, orange, yellow and green are an immediate assessment of the project for the user. The meaning of the colors is as it follows:

 2024 © Pizzato Elettrica S.r.l.

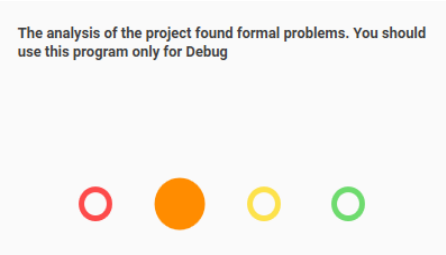


Result of the evaluation: RED

The program has errors that block the module programming.

The validation of the program wont' be available.

The program can't be sent to the module and C section will be hidden to the user.

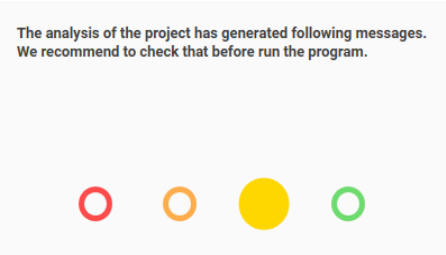


Result of the evaluation: ORANGE

The program compiles without errors but presents formal lacks that will have to be analyzed.

The validation of the program won't be available.

The program can be sent to the module.

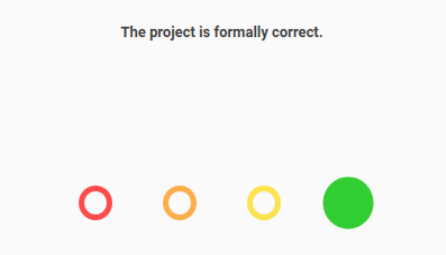


Result of the evaluation: YELLOW

The program has formal missing but this depends on the physical implementation of the electrical circuit or choices made by the user.

The validation of the program will be available as the customer may have already made the changes to the external circuit. Warnings will remain visible in the evaluation outcome.

The program can be sent to the module.



Result of the evaluation: GREEN

The program is formally correct (compilated and validated).

The program can be sent to the module.

⚠ The formal evaluation of the program has the sole purpose of helping the programmer to avoid errors or omissions and do not in any way replace the assessments and tests that must be done by the user (see also section 3.5).

B. Validation of the project: It formally conclude the design cycle. It is for this reason that the validation should be carried out by anyone other than those who conceived the program. Validation is closely related to the digital signature of the project. This, as noted, changes from time to time when they make changes to the project. It is important to understand that the validation is used to keep track of checks carried out by the operator before the official release of the program. It is for this reason that for every validation you have to insert a comment. This comment is stored in the validation log visible in the general section of the project (see paragraph 5.6.1). This log is stored in the project file.

 2024 © Pizzato Elettrica S.r.l.

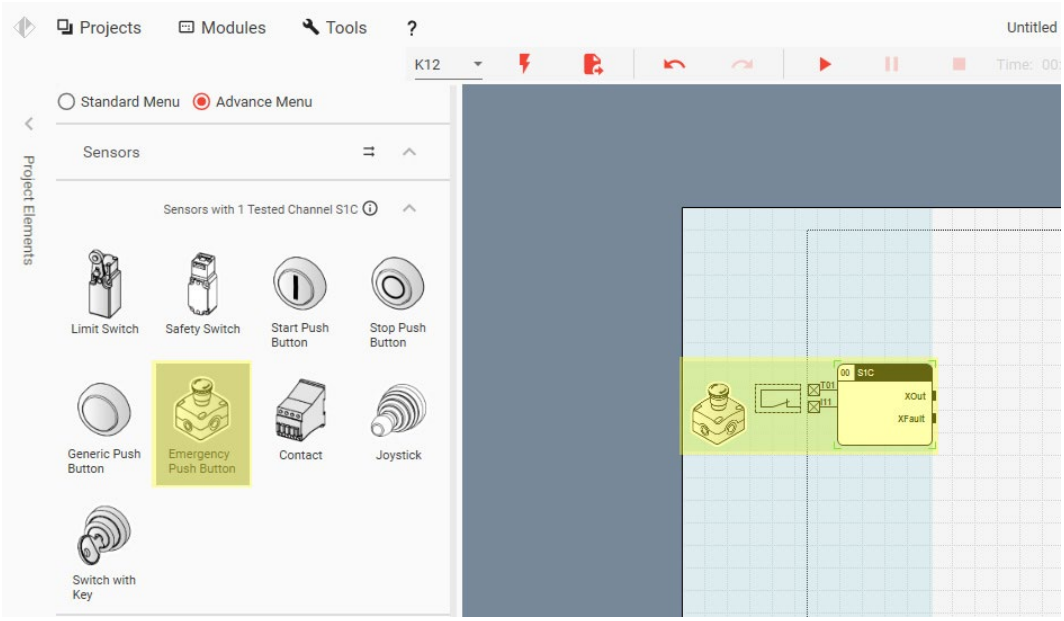
C. From this tab, you can send the program to the module connected via the USB port. The transfer sequence of the program activates the Monitor tab. This tab displays information about the status of the module and the transfer in progress (see 5.8). If no module is connected to the PC or there are errors in the program then this section will not be visible.

5.7.2 Sensors

Sensors

From the Sensor Panel you can select a sensor with the mouse and drag it to the desktop area. Inside Gemnis Studio this is equivalent to associate a specific safety feature to some module terminals, and to report the results of such a function on "connection points" that can be later used to transfer such data to other parts of the Desktop.

⚠ Choosing a sensor and the resulting type of electrical connections is crucial to the analysis of safety of machinery. Note that even by using the same external device (e.g. switch) can exist different types of sensor designed to handle that same device depending on how it is electrically connected to the module (eg. S1C or S1E) and that consequently the risk analysis is different.



Each sensor created in this way has a graphic area on the left with an image that represents the external physical device to the module and, when possible, a view of the internal configuration of contacts and how they are connected to the terminals of the module. To the right of the area is represented by a frame, the associated safety function and, inside the box, the parameters currently selected for the function.

The code of the safety function is highlighted in the upper part of the box whereas, to the left, is automatically created a progressive ID useful for discriminating sensors of the same type.

It is important to note that sensors "graphically different" can have the same underlying safety function. For example, the sensor that represents an emergency mushroom has the same safety function of a rope-switch for emergency stops, as far as electrical connection is concerned, since the two devices can be considered identical. The fact that there are two or more other graphic representations of the same safety

 2024 © Pizzato Elettrica S.r.l.

function is due to the goal of creating an environment that is as much as possible what is the real application.

The safety function associated with the sensor can produce many outgoing data, typically at least two: the standard output and standard error output of the function. The meaning and value of outputs from sensor to sensor can change. We recommend to read and fully understand the associated documentation (see later). However, in general, it there the assumption that the standard output is activated when the sensor is in his safe status and the safety function has not detected failures, on the contrary, when an error is detected, error output is active the standard output is deactivated.

> ⓘ Gemnis Studio is able to manage binary and numeric type information. The binary information (0,1) are identified by the prefix "X", the numerical 8 bit information by prefix "B" and the 16 bit information by prefix “W”.

The standard output of a sensor is typically of binary type and is then called "XOut", and similarly the standard error output is called "XFault".

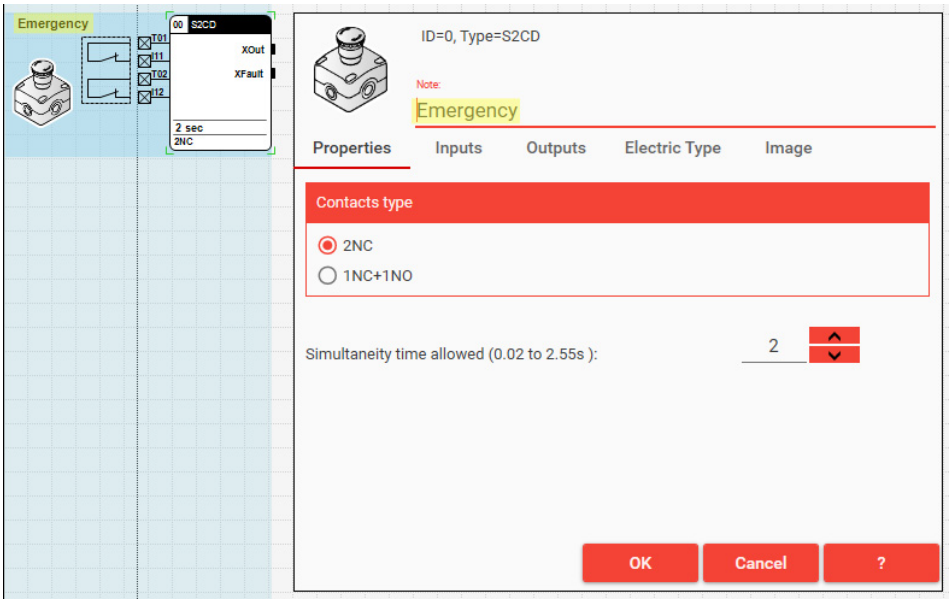
Sensor Status	Error function	XOut	XFault
any	Active	0	1
Not Safe	Not Active	0	0
Safe	Not Active	1	0

The sensor outputs are represented by small black boxes that represent the "connection points" of the sensor that are the areas from which you can start connections that transfer values of the outputs to other parts of the desktop.

Changes on sensors

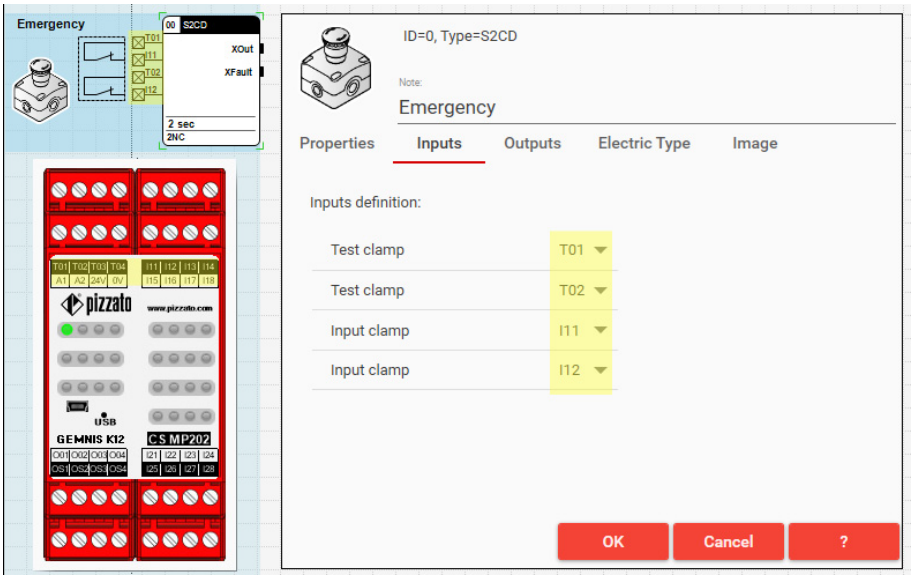
Every sensor is editable, and to do so, you can right click on the panel of the Sensor, and then click button Properties to access the Edit tab (or more briefly a simple double-click) or click on the button Delete to delete it from your desktop. Each sensor is equipped with a specific form but some customizations are common. For example, each sensor can be associated with a custom description that will appear in the graphics window.

 2024 © Pizzato Elettrica S.r.l.



Terminals

In each tab concerning sensors, in the Inputs section, are shown the terminals to which physical device is connected. The clamps are intelligently and automatically connected during the creation of the sensor but the user can still decide to change these associations by editing the data directly in the tab through a drop-down list. In modules list you may also have an immediate reference of where these terminals are physically present on the product (Modules → View modules).



Before you modify the data, you must however consider of the constraints imposed by the system, in particular that:

1. each input terminal (Ix) must be associated with only one sensor.
2. sensors that detect data from a pair of mechanical contacts must take Test Signals from sequential terminals and where the first clip is of odd and the second even index. For example, the pair T01-T02 is correct while the couple T03-T02 (first clamp even) or the couple T01-T04 (non-contiguous

 2024 © Pizzato Elettrica S.r.l.

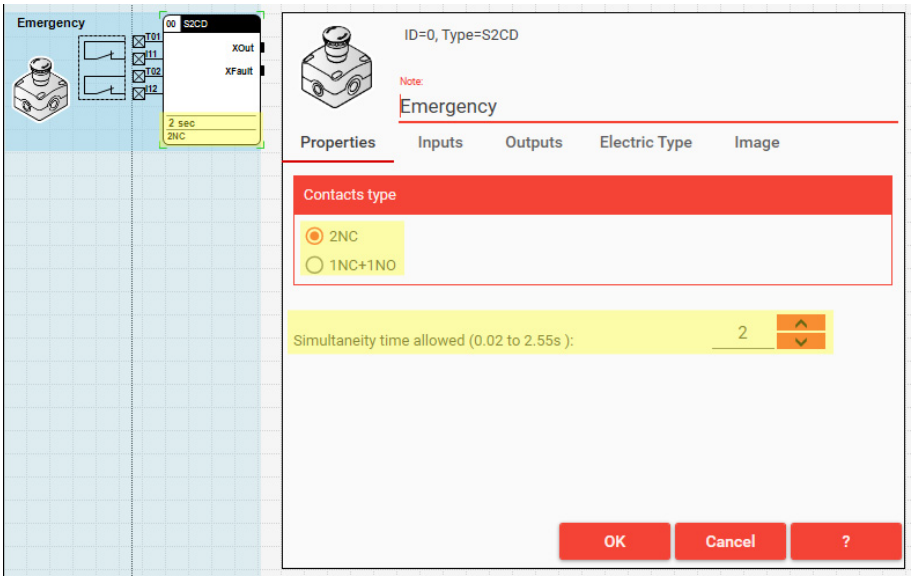
pair) are wrong. The data set by the user are checked during compiling.

3. some sensors, if present, introduce additional constraints that must be evaluated carefully by those who generate the program. For example, the use of devices that allow short circuiting between channels (such as safety mats) places restrictions on other sensors that don't allow short circuit because the second sensor would detect a fault. This type of problem can be solved either by dividing the test signals that by using diode networks external to the module. In these cases, it is up to the programmer to identify the appropriate solution.

Parameters

Some sensors are configurable: you can modify their standards behavior according to some specific characteristics of the real device. For example, some sensors allow you to vary the standard scheme of contacts (from 2NC to 1NO+1NC).

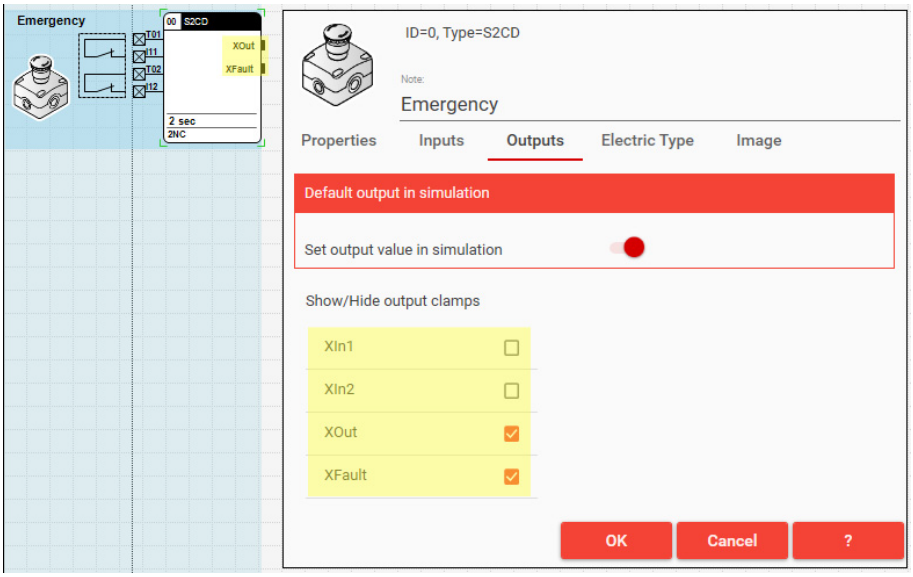
Sensor parameters are listed in the sensor and you can modify them. The parameters and the value set are highlighted graphically in small rectangles that are at the base of the sensor.



 2024 © Pizzato Elettrica S.r.l.

Sensor outputs

In addition to the typical standard outputs XFault and XOut each sensor can have additional outputs that, being less common use are hidden during creation. These outputs can be made visible by changing their state, in sensor tab, in outputs section.



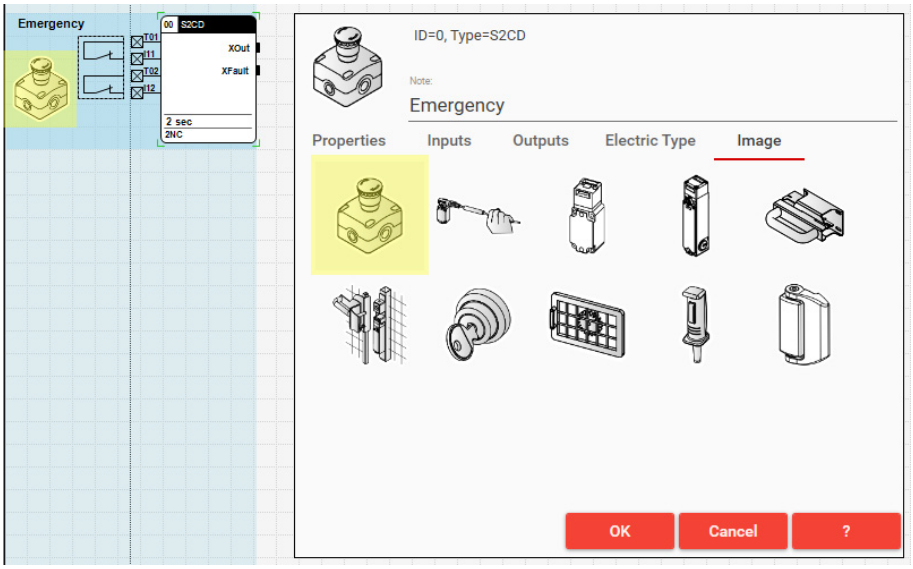
On this tab you can also set the default behavior of the sensor during simulation phase or the State in which it will be at the start of each session.

Electric sensor types

This tab shows the image that defines the type of device. See also paragraph 5.7.2.

Image of the sensor

On this tab, you can select the image associated to the sensor. In fact, sensors, even if they have the same electrical typology, can have different "physical forms". Depending on the type of sensor we have preloaded several images that represent the most commonly used safety devices with that type of sensor.

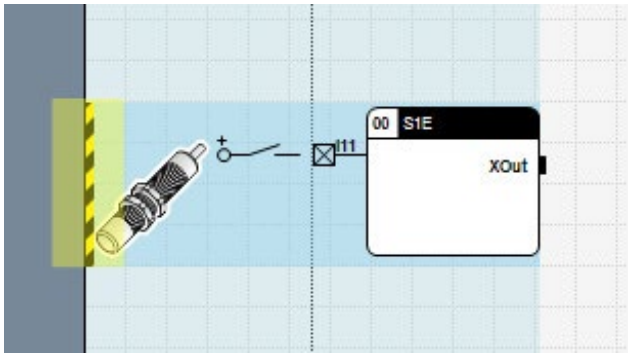


 2024 © Pizzato Elettrica S.r.l.

Safe sensors and "unsafe" sensors

Some real devices once connected to the module are checked very thoroughly and the safety function of the module is capable of detecting many types of failures. Other sensors, for their intrinsic nature, have failures which cannot be controlled by the module. These devices can also be connected to the module but it is up to the programmer to evaluate in depth the possible faults of such devices and their possible consequences for the functions performed by the module (see documentation). To remind the programmer and make clearer their unsafe nature they are reported through a colorful striped black and yellow sidebar (see figure).

Note that "not safe" nature is not due only, this case, to the device itself but also how it is connected to the module. Then an electric type sensor S1E (see S1E Sensor) can be for example an inductive sensor but also a safety switch with positive opening. In this case is the type of electrical connection that does not allow us to exclude certain types of failure, such as links, and then, for this matter, this device is highlighted.



Definitions

In the definition of the sensors the following terms are used:

- *Electric Type*: classifies the electrical nature of the device. The images that represent the electrical nature of the devices should be interpreted as follows:
    - the box represents the device and, on the inside, there is an indication of schematic electricalcircuit within the device, typically mechanical contact or transistor.
    - the "+" symbol represents the voltage present on Terminal A1 module (typically + 24VDC)
    - the arrow symbol represents the output or the outputs of the device and it will be linked to module inputs
    - The symbol of the dashed line with two arrows at the ends is a dependency between the parts of the device. The dependency can be mechanical in nature (contacts mechanically linked) or functional (transistors that are driven simultaneously) and indicates a constraint that the sensor uses to determine failure of non-coherence between signals.
    - The symbol ‘T’ followed by a square wave indicates the Test signal generated by a clamp. If the test signals used by the sensor are two, they are typically represented by "T1" and "T2" with square waves in phase opposition.
    - The symbol "Tc" indicates a fixed temporal simultaneity between two fixed device channels
- *Examples*: a list of real devices that, according to their electrical connection, can be traced to the typeof the considered sensor. The examples are not intended to be exhaustive and are meant only to demonstrate visually, inside of Gemnis Studio, the type of product that the user has physically connected to the module. It is not the "shape" of the device but that such a device is connected as

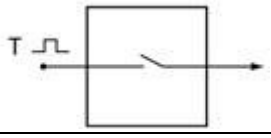
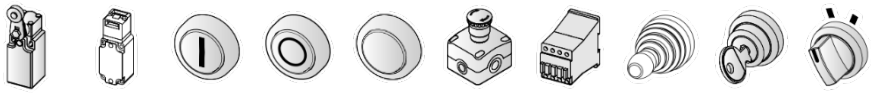
 2024 © Pizzato Elettrica S.r.l.

required by the type of sensor.

- *Connections*: indicates the number of input terminals and Test signals needed to check the safety device.
- *Parameters*: Indicate the possibility of sensor configuration to adapt to the physical nature of the Safety Device.
- Parameters-*Types of Entrances*: the term 2NC means that the device has 2 normally closed electrical contacts in safety conditions which open both in the situation of danger. In the case of electronic devices, it means that the device is equipped with two electronic active outputs both closed in safeconditions and both off in the situation of danger. With 1NC + 1NO term means that the device hasa closed and an open contact safely exchanging their position in a position of danger. In the case ofelectronic devices, it means that the device has two outputs and an electronic active safely off thatreversed the polarity in the situation of danger.
- *Parameters – maximum time of simultaneity*: sensors with multiple inputs, this parameter identifies the time limit within which you must bring inputs into a position considered coherent, depending on the chosen Sensor. Typically, the time of contemporaneity is needed in mechanical devices to verify the correct positioning of contacts within a time limit to avoid problems due to rebounds or "inaccurate activations " by users of the machine.
- *Outputs*: lists the name of connection points made available by the sensor and their function
- *Detectable device failures:* Lists the possible faults that the sensor, depending on the type of device, isable to detect.
- *Behavior at startup:* Indicates the behavior of the sensor depending on the data at startup.

 2024 © Pizzato Elettrica S.r.l.

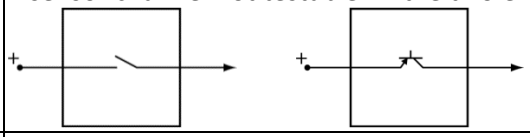
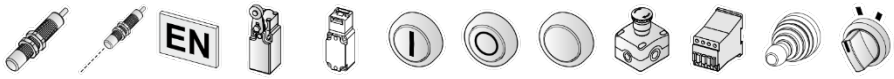
5.7.2.1 S1C Sensor

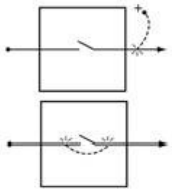
Electric type	Sensor with 1 tested channel 
Examples	Various mechanical devices with 1 NC contact (e.g. buttons, switches, etc.) 
Connections	1 Input 1 test signal
Parameters	None
Outputs	XOut: contact State if there are no errors due to short-circuiting XFault: active if a short circuit is detected
Detectable device failures	Short circuit to positive, short circuit to other test signals
Behavior at startup	XOut indicates the State of the entry

⚠ This sensor detects a significant number of external electrical fault to the device but not internal ones such as the internal short-circuit (contact welding). If this sensor is used to generate a start-up command is recommended to use an additional detection function (controlled start) to avoid that a contact welding causes unexpected start-ups.
In case of short circuit to ground the failure is not detected by the Sensor (which however interpret the contact as open) but from overcurrent monitoring of the module.
In the case of safety devices (pushbutton for emergency stops or safety switches with a single contact NC) it is important to use positive opening devices in order to exclude the possibility of contacts welding. In any case, with single-channel devices, there are important limitations on the highest levels of safety reachable (see safety standards).

 2024 © Pizzato Elettrica S.r.l.

5.7.2.2 S1E Sensor

Electric type	1 sensor channel not testable. In the two examples. 
Examples	Various types of electronic devices with PNP output (inductive sensors, photocells, etc.) or mechanical devices with 1 touch connected directly to the positive (e.g. buttons, switches, etc.) 
Connections	1 Input
Parameters	None
Outputs	XOut. Indicates the value of the incoming connection
Detectable device failures	None
Behavior at startup	XOut indicates the State of the entry

⚠ This sensor alone is not able to detect the device's electrical fault such as short circuit to positive or toward mass or functional breakdowns as the welding of contact or, in the case of electronic outputs, the output transistor short circuit. The output of this sensor should not be used directly for safety-related functions. Can be used for example for controlling the Start button but only if linked to the use of an additional analysis function, for example signal front within the Gemnis code, or, however, of a function that does not allow	Possible faults 

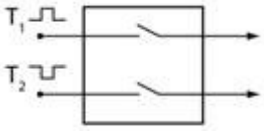
Typically, devices that are attached to this sensor are not used alone and the safety function that they perform is functional. For example, most optical sensors can be used to create a muting function. This function is based on a specific sequence of activating and deactivating of devices in a considered time. A failure on one of the devices would then intercepted not so much from the safety function of the sensor but by the application program.

There are also other cases where this type of devices can be used, for example by implementing procedures for periodic testing (according to the risk).

To remind the programmer and make clearer their unsafe nature they are reported through a colorful striped black and yellow sidebar.

 2024 © Pizzato Elettrica S.r.l.

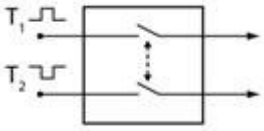
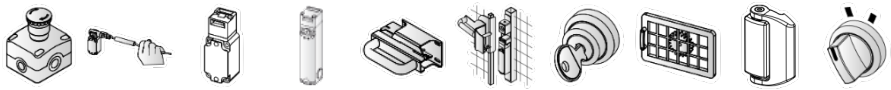
5.7.2.3 S2CI Sensor

Electric type	2 independent channels sensor tested 
Examples	Various mechanical devices with 2 contacts (2NC or 1NC + 1NO) mechanically independent e.g. safety magnetic sensors, pairs of switches (also of different type) mechanically connected to the same shelter but that can act at different times. 
Connections	2 Inputs 2 test signals
Parameters	2NC or 1NC + 1NO (intended in safe conditions)
Outputs	XOut: active if the inputs are in the expected and there are errors XFault: active if a short circuit is detected XIn1: input Value 1 XIn2 2 input Value:
Detectable device failures	Short circuit to ground one or both channels, short circuit to positive of one or both channels, short circuit between the two channels or other test signals
Behavior at startup	When both channels are in consistent state XOut is active

⚠ This sensor detects a significant number of external and internal electrical fault in the device. Internal short-circuit between channels are detected. The internal short-circuit of individual channels (contacts welding) or rupture are not detected. Due to the infinite contemporaneity time this Sensor is not able to distinguish a single break from working properly. In the case of a single break the dual-channel structure of this sensor allows however to carry out the safety function. The fault is detected on first demand.

 2024 © Pizzato Elettrica S.r.l.

5.7.2.4 S2CD Sensor

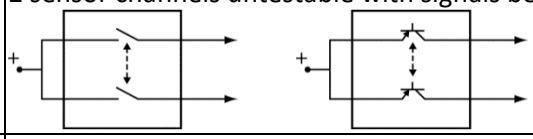
Electric type	2 dependent and tested channels sensor 
Examples	Various mechanical devices with 2 contacts (2NC or 1NC + 1NO) mechanically dependent such as pushbuttons, rope switches for emergency stop, safety switches, safety hinges 
Connections	2 Inputs 2 test signals
Parameters	Input type: 2NC or 1NC + 1NO (intended in safe conditions) Maximum time of contemporaneity: selected between 0.02 and 2.55 seconds
Outputs	XOut: active if the inputs are in the expected state and there are no errors XFault: active if there is a short circuit or a lack of simultaneity XIn1: input value 1 XIn2: input value 2
Detectable device failures	Short circuit to ground one or both channels, short circuit to positive of one or both channels, short circuit between the two channels or other test signals, welding or rupture of the individual channels.
Behavior at startup	When both channels are in consistent state XOut is active

This sensor practically detects all electrical fault. The internal short circuit is detected between the device and the internal short-circuit of individual channels (contact welding) or failure. Having a finite contemporaneity time, the Sensor can distinguish a fault by detecting the inconsistency between inputs, as soon as the maximum time allowed is exceeded.

Note that the "dependence" of contacts does not imply a physical link between the contacts but is simply a logical dependency. Within this category are not therefore only devices with two contacts mechanically interconnected but also devices that, for example, as they are mechanically connected to the same shelter, then must move simultaneously.

 2024 © Pizzato Elettrica S.r.l.

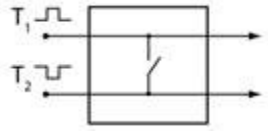
5.7.2.5 S2ED Sensor

Electric type	2 sensor channels untestable with signals between them dependent 
Examples	Electronic devices of various types with a double PNP output (optical barriers, RFID sensors, sensors with double output OSSD type, etc.) or mechanical devices with 2 contacts connected directly to the positive (e.g., emergency buttons, switches with two contacts, etc.) 
Connections	2 Inputs
Parameters	Input type: 2NC or 1NC + 1NO (intended in safe conditions). Maximum time of contemporaneity: selected between 0.02 and 2.55 seconds
Outputs	XOut: active if the inputs are active and consistent and there are no errors of contemporaneity XFault: active if there is a timeout for the maximum time of simultaneity XIn1: Value for input 1 XIn2: Value for input 2
Detectable device failures	Inconsistency of the outputs of the device when it is higher than the set time
Behavior at startup	If both channels are in consistent state XOut is active

⚠ This sensor alone is not able to detect some electrical fault external to the device such as short circuit between channels. Short circuit to positive or mass of a single channel is instead detected as not inconsistent signals for an infinite time.
This type of sensor is widely used but may be necessary additional protection measures in the wiring. If the device is equipped with an electronic safe output (OSSDS type for example) this implies that in case of short circuit to ground or positive to a canal or in the case of short circuit between channels the device disables both its outputs and therefore detection of individual failures is however guaranteed. In the case of devices that do not have controlled outputs are typically necessary additional measures such as physical separation of cables.

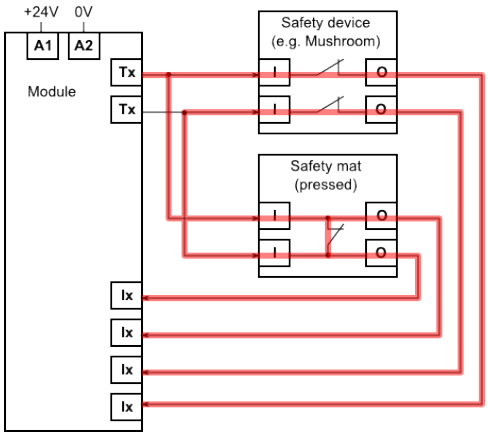
 2024 © Pizzato Elettrica S.r.l.

5.7.2.6 SSM Sensor

Electric type	2 sensor channels always tested with closed short circuit between channels 
Examples	Safety mats or sensitive edges with 4 wires technology 
Connections	2 Inputs 2 test signals
Parameters	None
Outputs	XOut: active if there is short circuit and there are errors XFault: active if there is a short circuit to positive or the wires are broken XIn1: input Value 1 XIn2 2 input Value:
Detectable device failures	Short circuit to ground one or both channels, short circuit to positive of one or both channels, channel break.
Behavior at startup	When both channels are in consistent state XOut is active

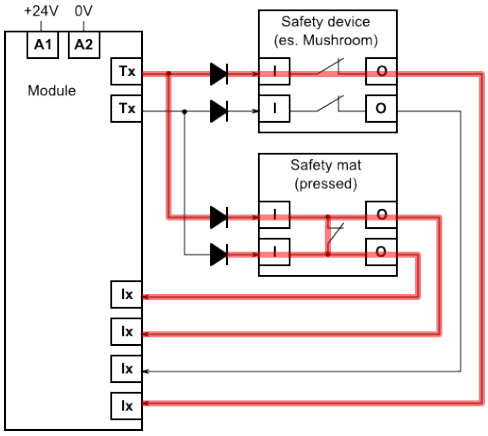
This sensor detects virtually all electrical fault. It detects the short circuit to ground, to the positive and the disconnection of wirings.

You must pay attention to the fact that this device, accepting short-circuit, when connected to the same Test signals of other devices that do not admit it, may led to detect non-existent short circuits on those other devices (see figure).



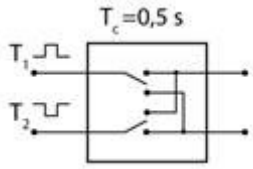
For this reason, it is necessary to maintain the Test signals for this kind of devices separate from the other, selecting modules equipped with multiple Test signals. Alternatively, you can wire a network of external diodes (see figure).

 2024 © Pizzato Elettrica S.r.l.



⚠ Due to the mechanical "large" nature of these devices (especially in the case of multiple safety mats connected in series), it is necessary to pay special attention not to exceed the maximum electrical capacity allowed for inputs channel of the module (see technical specifications).

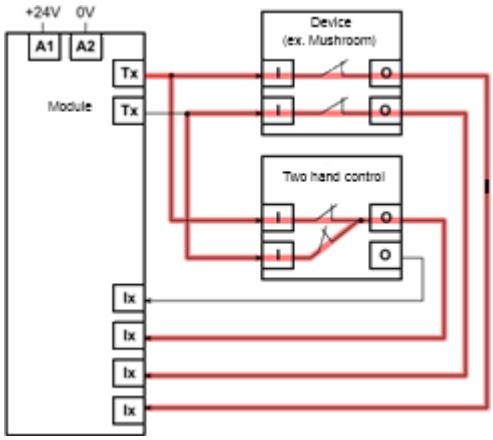
5.7.2.7 STHC Sensor

Electric type	Sensor with 2 channels tested that you can cross 
Examples	Category IIIC bimanual commands 
Connections	2 Inputs 2 test signals
Parameters	None
Outputs	XOut: active if both contact commute from the rest position to work within 0.5 seconds. XFault: active if there is a short circuit to positive XIn1: input Value 1 XIn2: input Value 2
Detectable device failures	Short circuit to ground of one or both channels, short circuit to positive of one or both channels.
Behavior at startup	XOut is zero, even if the contacts are in working position, you must release your contacts and then press.

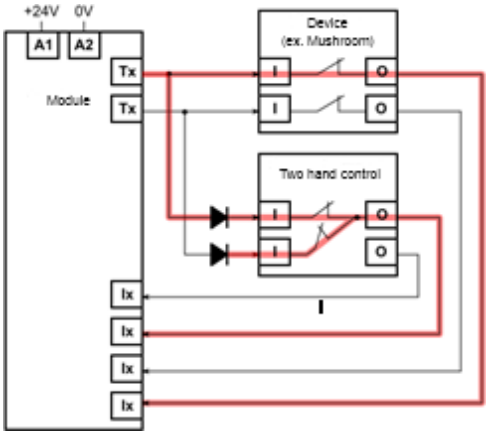
Since it is admitted that contacts can cross, some failures cannot be detected directly by the sensor. For example, breaking or welding a contact keeps the device in a State still normally possible. In any case, in these conditions the standard output of the sensor is inactive.

You must pay attention to the fact that this device, admitting short circuits when connected to the same test signals of other devices that do not admit it, instead, can detect short circuits that may not exist in such other devices (see figure).

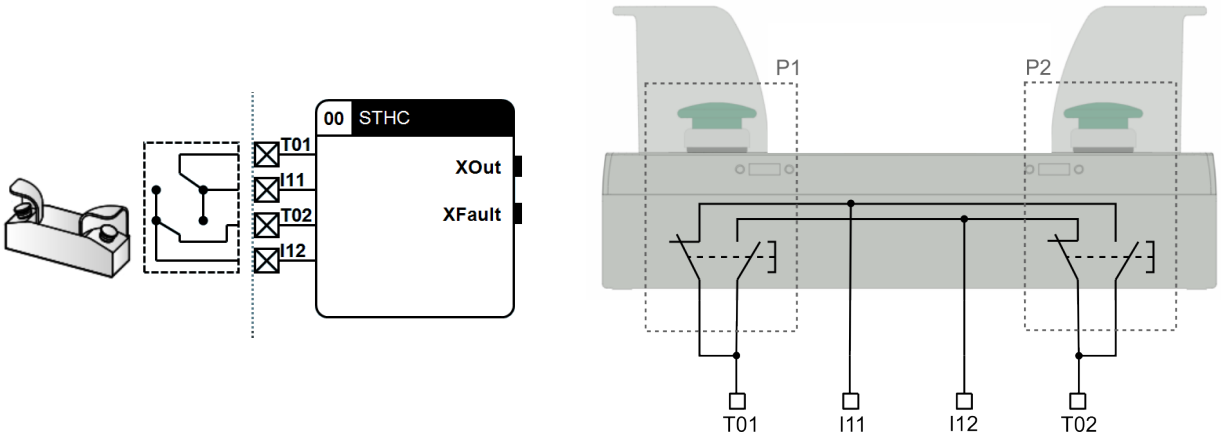
 2024 © Pizzato Elettrica S.r.l.



For this reason, it is necessary to keep the test signals for this kind of devices separated from the others by choosing modules with multiple test signals. Alternatively, you can insert a network of external diodes (see figure).

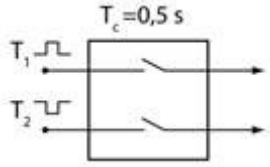


The following pictures represent the correct connection of a two-hand control device in accordance with the STHC sensor block. The status of the contacts is intended when the buttons are not operated.



 2024 © Pizzato Elettrica S.r.l.

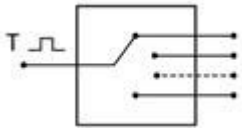
5.7.2.8 STHA Sensor

Electric type	Sensor with 2 tested channels not crossable 
Examples	Category IIIA commands bimanual 
Connections	2 Inputs 2 test signals
Parameters	None
Outputs	XOut: active if both switch contact from the rest position to work within 0.5 seconds. XFault: active if there is a short circuit to positive XIn1: input Value 1 XIn2: input Value 2
Detectable device failures	Short circuit to ground of one or both channels, short circuit to positive of one or both channels, short circuit between the two channels or other test signals
Behavior at startup	XOut is zero, even if the contacts are in working position, you must release your contacts and then press them

Due to the mechanical structure of this type of devices some kinds of internal failure cannot be directly detected by the sensor. For example, breaking or welding of a contact keeps the device in a state still normally possible. In any case, in these conditions the standard output of the sensor is inactive.

 2024 © Pizzato Elettrica S.r.l.

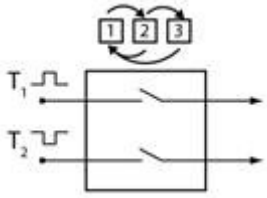
5.7.2.9 SSE8 Sensor

Electric type	Sensor with 2 to 8 tested channels that cannot cross and with only one active at a time. It is accepted that for a short time the device is inconsistent (for example during the switch-over from a contact to another). 
Examples	Selector switches or modal selector from 2 to 8 positions 
Connections	From 2 to 8 Inputs 1 test signals
Parameters	Number of inputs (from 2 to 8)
Outputs	BOut: Indicates with a value from 1 to 8 the active entry; the value is 0 in case of error XFault: active if there is a short circuit to positive or between channels or if the selector does not reach a stable position within 0.5 seconds
Detectable device failures	Short circuit to ground all channels, short circuit to positive of all channels, short circuit between any channel, shorting any channel to other test signals, disconnection of test signal, disconnect of any channel, mechanical breaking of selector with open contact.
Behavior at startup	BOut indicates the active input

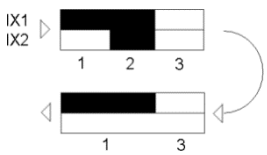
With this sensor can practically detect all types of electrical failures of this type of devices and also some mechanical breakdowns as breaking of the contact. Some electrical fault may not be detected immediately, such as a disconnecting of an input wire, but they will be detected the first time that function is requested. It is allowed that, with "slowly" operated selector, this Sensor may create incorrect contact combinations (all contacts open or closed two contacts simultaneously) but only for a maximum time of 0.5 seconds. During this phase the output does not change, i.e. Bout is stable until a correct position of the selector is reached.

 2024 © Pizzato Elettrica S.r.l.

5.7.2.10 SPE3 Sensor

Electric type	Sensor with 2 tested channels which can't cross and that must follow a precise sequence of activation/deactivation composed by three states: Rest, Work, Stop. See examples. 
Examples	Enable pushbuttons, or safety footswitches 
Connections	2 Inputs 2 test signals
Parameters	Type 2NC inputs (means in work position).
Outputs	XOut: indicates whether the button is in Work position XStop: indicates whether the button is Stop position (strongly tighten) XFault: short circuit to positive or wrong state XIn1: input Value 1 XIn2: input Value 2
Detectable device failures	Short circuit to ground one or both channels, short circuit to power supply of one or both channels, short circuit between the two channels or other test signals
Behavior at startup	If the device is in Work position XOut remains at 0 until it goes first to the Rest state. XStop behaves normally.

The SPE3 sensor provides control of an electrical device that has the same electrical connections of others but admits only a specific sequence between three states defined Rest, Work and Stop as shown in figure. When the button is released, the device is in a Rest state (1). When the button is pressed but with normal pressure the device is in Work state (2). When the button is strongly pressed the device is in Stop state (3).



The device is built in such a way that they are possible only following state transition:

1→2, 2→1, 2→3 and 3→1

In practice this means that after being entered in the Stop state the device cannot return immediately to Work state but must first pass in Rest state, to prevent an unexpected restart of the machine.

⚠ This sensor detects a significant number of external electrical fault in the device but not all internal ones such as the internal short-circuit (contact welding). The particular structure of the sensor which provides for a specific sequence of States allows to identify even some internal faults.

 2024 © Pizzato Elettrica S.r.l.

⚠ Some 3 stages enabling pushbuttons have contacts made in such a way that states 1 and 3 are electrically indistinguishable. For example, in the case where the sensor has two open contacts both in state 1 and 3. This type of enable pushbuttons must be managed with the S2CD (see Sensor S2CD Sensor).

5.7.2.11 SUSB Sensor

Electric type	Integrated sensor
Examples	USB port of the module 
Connections	USB port of the module
Parameters	NonSafe or Safe Protocol
Outputs	XNoUsb: indicates a missing USB connection Out (n): 8 outputs of user definable type (only with Protocol = Safe)
Detectable device failures	Disconnect the USB cable
Behavior at startup	

This sensor is a "virtual" one because it does not check external devices but is typically used by an external controller (PC or PLC) to enter data into the module in safety manner. Input data are presented on 8 different connection points whose type (X, B or W) is user-defined.

Not Safe Protocol

If the sensor is used with parameter "Not Safe" it presents only the "XNoUsb" connection point, a type X connection that has a value of 1 if the USB port is not connected. This output is used usually to activate some output signal of the module in order to warn the user that the USB cable is disconnected.

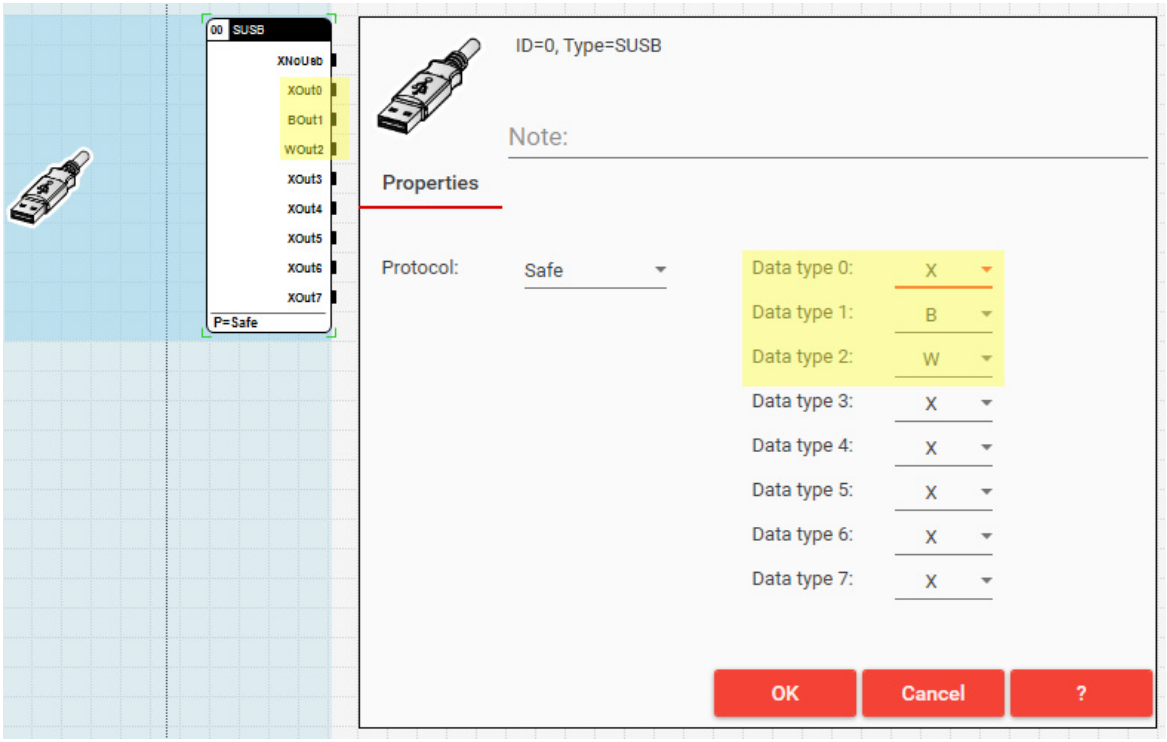
⚠ The XNoUsb is not a safe signal and should not be used to enable safety outputs.

Safe Protocol

If the sensor is used with parameter "Safe" it introduces, in addition to XNoUsb, other 8 connection points, named Out0-Out7, of user settable type. Depending on the user-defined type, outputs change names. For example, if the 0 index connection point is set to type X, its name will become "XOut0", while if type B is chosen, its name becomes "BOut0".

In the figure, the outputs having index number 0,1 and 2 have been respectively set of type X, B and W. Consequently, connection points have assumed the name of "XOut0", "BOut1" and "WOut2"

 2024 © Pizzato Elettrica S.r.l.



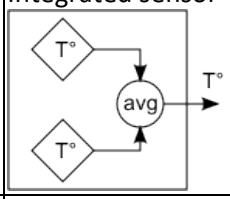
At run-time through the writing protocol (see Writing data into the module) an external controller can write data on these variables that become for all intents inputs for the Application Program.

This type of sensor is used in case you intend to use variable numeric values, e.g. for different timings according to the operational status of a specific machine or according by used tools.

⚠ This functional block is a powerful tool that should be used carefully by the programmer. It is not recommended to use this Sensor to allow safety systems bypass.

 2024 © Pizzato Elettrica S.r.l.

5.7.2.12 STIN Sensor

Electric type	Integrated sensor 
Examples	Double temperature sensor integrated in the module 
Connections	None
Parameters	Mode of representation of the temperature
Outputs	BTemp: the average value of the temperature detected XFault: excessive temperature difference between the two sensors
Detectable device failures	Rupture of an internal temperature sensor
Behavior at startup	

This sensor is intrinsic because in each module of the series Gemnis there are two integrated sensors of temperature controlled independently by the two processors of the system.

The value of the average temperature detected by the two sensors is disclosed in connection BTemp, using unit of measure provided by the parameter that is:

- Increased to 100 ° Celsius (for not having to deal with the sign of the sub-zero temperatures)
- Degrees Fahrenheit unsigned.

Since Gemnis does not handle numeric values with sign it follows that the representation of degrees Celsius for temperatures below zero would be impossible. For this reason, it was intended to represent the internal temperature of the module in degrees Celsius incremented by a fixed value of 100 so that all permissible values are positive. For example, an internal temperature of 40 ° C will be represented as 140 °, while a temperature of -10 ° C will be represented as 90 °.

The use of this sensor is suggested as a warning signal, before the temperature reaches values in the context that may lead to the arrest of the module for under-or over operating temperature (see technical data in the form to the admitted range).

The accuracy of the internal temperature sensors is ± 2 ° C.

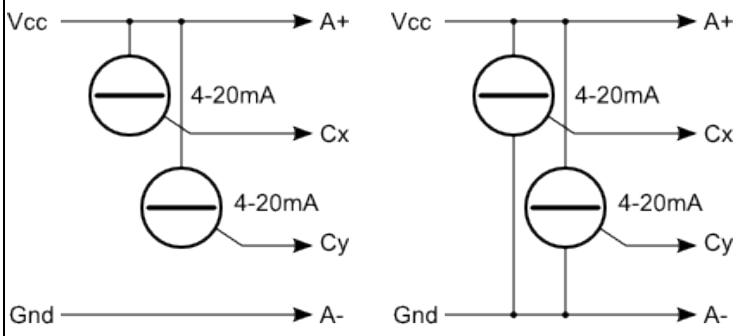
The indicated temperature is the average temperature of the two internal sensors.

If the difference between the two internal sensors exceeds 10 ° C due to high thermal imbalances or due to the rupture of one of the two internal sensors, the sensor STIN activates the output XFault.

⚠ You can’t use this sensor to detect safely temperatures outside the module, not only outside but also on the context of the devices or specific areas within the panel. The temperature measured inside the module depends on many factors including the physical location of the module in the panel, from devices that work with it and the heat they generate. In a panel with good ventilation (in order to standardize the internal temperature) and after experimental analysis this sensor can provide an indirect indication of the temperature inside the panel but with precision, response time and value to be determined on a case by case basis and in any case not guaranteed by the manufacturer of the module.

 2024 © Pizzato Elettrica S.r.l.

5.7.2.13 SAN1 Sensor

Electric type	Control a pair of analog sensors with 4-20 mA output in both 2-wire 3-wire version.  The power supply of the sensors can be taken in parallel with the power supply of the module (Vcc = A1, A2 = Gnd) or by separate power supply.
Examples	Redundant control of a single analog parameter such as temperature, current, vibration, etc.. 
Connections	See the examples. A + means the power supply terminal and A- the ground terminal of the board to which the 4-20 mA sensors are connected. The nomenclature of A + and A- is dependent on the module (see General Hardware Characteristics).
Parameters	Mode of representation of the mean value in Byte or Word. Maximum permissible difference between the values measured by the two channels (coherence) Maximum allowable non-coherence time Minimum and Maximum current allowed for the two 4-20 mA sensors
Outputs	Bout or WOut: the mean value of the two sensors XFault: Active in case of error BFaultCode: In case of error indicates the numeric error code BOut1 (optional): The current value detected by the first sensor BOut2 (optional): The current value detected by the second sensor
Detectable device failures	Over or undercurrent of both sensors 4-20 mA or broken devices (short-circuits or openings) or electrical connections. Inconsistency between the incoming data from the two sensors, or a sensor decalibration or physical removal of a sensor by the magnitude in phase of detection.
Behavior at startup	On the module startup, the 4-20 mA sensors need to be active and able to provide consistent data with the set parameters.

ⓘ This type of sensor is displayed in Gemnis Studio only on modules with analog inputs and with terminals of type "C".

This type of sensor was designed to control analog inputs of type 4 -20mA redundant. The objective is to detect a physical quantity using a pair of devices, and calculate the average value as the representative value of the physical quantity. At the same time, we want to verify that the values of the two sensors are

 2024 © Pizzato Elettrica S.r.l.

consistent, namely that the two values consistently maintain a difference between them below a set percentage.

The lack of coherence of the two sensors allows to identify faults in the two devices such as:

- Non- operating (device short-circuited or disconnected or stuck on a fixed value)
- Prong of the devices, namely loss of linearity of conversion (identifiable when the difference between the two values exceeds the maximum set percentage)
- Change of the response time of the devices (which leads to an inconsistency of signals)
- Some mechanical problems, such as the physical detachment of one of the two devices from the support that connects it to the environment to be measured.

Defined IS1 and IS2 the value of the input currents of the two channels, this sensor reports in the output Bout (or WOut) the average value of IS1 and IS2. The user can choose whether to express the average value by an output Bout of type byte (8 bits) expressed in units of 0.1 mA or alternatively through an output WOut of type Word (16 bits) expressed in units of 0,01 mA. So, for example, a value of 100 in Bout indicates that the average current in the two sensors is equal to 10.0 mA while a value of WOut equal to 1205 indicates that the average current is equal to 12.05 mA.

The output type is BOut of default and requires fewer resources of the module, while the output of the module type WOut is preferred when we demand greater detection accuracy.

ⓘ The output BOut imposes a maximum value detectable, constrained by the capacity of the variable representation of type B, equal to 25.5 mA. Average values in excess of this limit will still be represented as 25.5 mA.

ⓘ The type output WOut allows instead represent values greater than 25.5 mA, but in this case is assessed in the documentation associated with the module the range of values for which it is guaranteed the accuracy and / or the linearity of conversion.

The user can set the maximum allowable percentage difference between the two values IS1 and IS2, called ID%, and the maximum time TF in which this condition is permissible. The difference is calculated with reference to the signal IS1. When IS2 outside the range IS1 ± ID% you have inconsistency. The sensor software will set the output XFault when IS1 and IS2 are constantly inconsistent for a time greater than TF (adjustable from 0 to 2.55 seconds).

The ability to set a time inconsistency is very useful to avoid that spurious peaks of the values of IS1 or IS2 (typically at startup or during shutdown of a process) bring the system error condition. It should however be borne in mind that the introduction of a filter time also increases the response time of the module to a fault. The two values must therefore be as set in the function of the physical characteristics of the equipment in question and of the sensors connected to it.

Ways of calculation of the inconsistency

In Mode 0 the two analog devices with 4- 20mA output are assumed with identical characteristics in output. The differences between the output values of the two sensors will be due only to the different manufacturing tolerances and to the different positioning of the devices compared to the magnitude in the measurement.

Sometimes it may not be easy to identify, when debugging, if the input sensors behave as expected. By activating the outputs BOut1 and BOut2 (or WOut1 and Wout2) you can read the value of the two input devices.

 2024 © Pizzato Elettrica S.r.l.

If instead you use devices with different signal linearization, or if you plan to compensate some systematic differences between the sensors is possible, by setting the Mode 1, translate or amplify the signal from the sensor 2 through two constants. The first constant allows to add to the value IS2 a fixed value selectable in the range ± 20.00 mA, while the second constant allows to attenuate or amplify the signal up to 4 times.

In Mode 1, if you enable debugging output BOut1 and BOut2 (or WOut1 and WOut2), please note that BOut1 is identical to the value of the first input, while BOut2 represents the value of the second input after the translation and amplification of the signal.

Maximum and minimum current

The sensor software also provides the ability for the user to set a maximum and minimum current such that if any of the two 4-20 mA devices out of admitted range of work, it is set as an error. Typically, the thresholds are set slightly outside the normal range (minimum current of 3.8 mA, maximum of 21 mA), but the user can adjust at will depending on the installed devices. Exceeding the maximum or minimum threshold set immediately the error condition, and doesn't apply the filter time.

List of error codes for the sensor

This sensor allows several types of error detected. For simplicity, it is preferred to have a single output that is activated in the event of a fault (XFault) and return in another variable the error code (BFaultCode). In this way, the expert user can go to understand that error occurred and possibly integrate a different error handling routine from case to case. It should be noted that to the disappearance of error conditions, the outputs related to the errors are resetted.

The error codes available are:

BFaultCode	Description
0	No errors
1	Average current first sensor below the minimum set
2	Average current first sensor exceeds the set maximum
3	Average current second sensor below the minimum set
4	Average current second sensor exceeds the maximum set
10	Data collected from the two sensors inconsistent
100	No power on terminals A, or rupture of the internal ADC converters
200	Internal error of the board

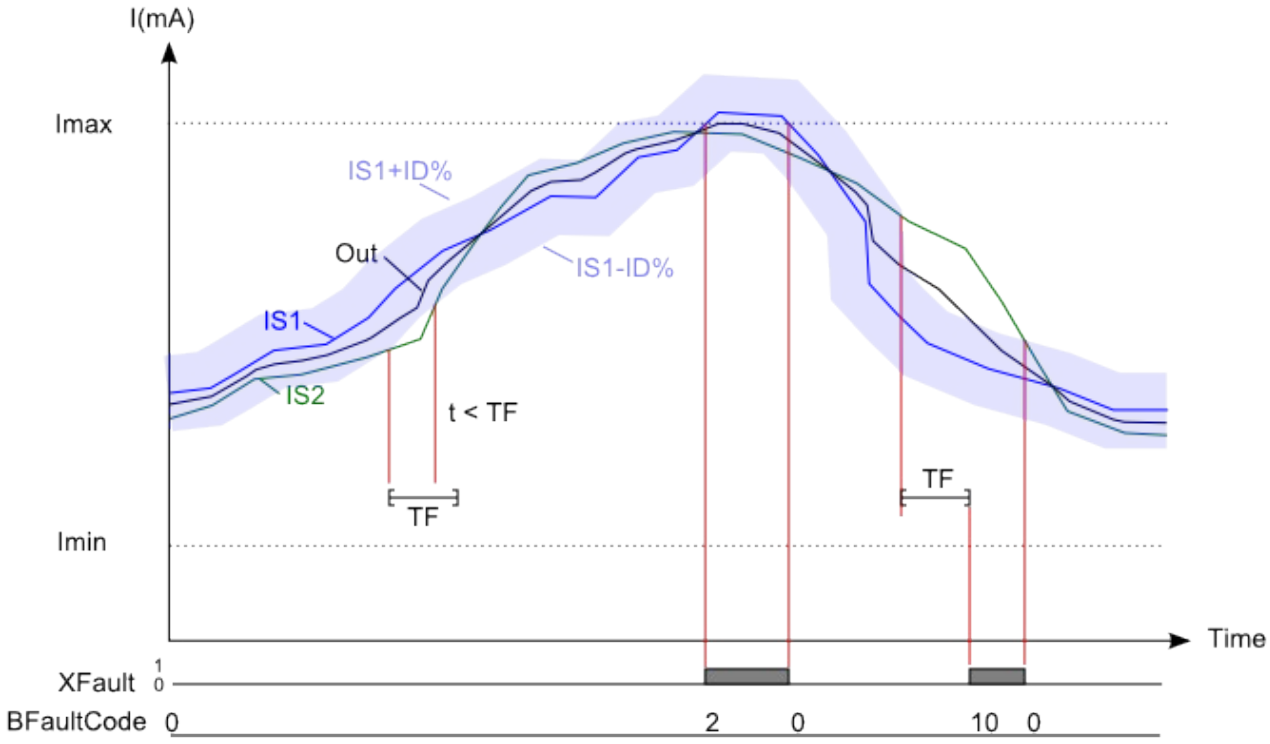
In case of error with code 100 or 200, the read data may not make sense, therefore the output value from the function of the sensor is set to the max value allowed by the data type or rather 255 in case of Bout and 65535 in case of WOut.

The basic principle is that an error is prevalent with higher value on those with lower value. In case of failure of the supply of the sensors, for example for detachment of the connector (terminals of type A), it indicates the error 100 and not the error 1 or 3.

 2024 © Pizzato Elettrica S.r.l.

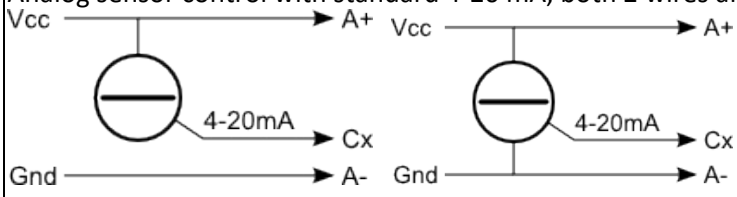
Example

In the chart below, you can see a possible example of the behavior of the sensor as a function of two signals IS1 and IS2.



 2024 © Pizzato Elettrica S.r.l.

5.7.2.14 SAN3 Sensor

Electric type	Analog sensor control with standard 4-20 mA, both 2 wires and 3 wires.  The sensor power supply could be taken in parallel from the module power supply (Vcc=A1, Gnd=A2) otherwise with a separate power pack.
Examples	Analogic parameters monitoring, like temperature, current, vibration, etc. 
Connections	See examples. A+ stays for the power supply terminal and A- stays for ground terminal of the board where the 4-20 mA sensor is connect. Nomenclature of A + and A-depends on the module (see General Hardware Characteristics).
Parameters	Average value representation mode, in Byte or Word. Minimum and Maximum current allowed for the 4-20 mA sensor.
Outputs	BOut or WOut: detected value XFault: Active in case of error BFaultCode: error number when XFault is activated
Detectable device failures	Overcurrent or undercurrent of the 4-20 mA sensor, or device damages (short circuit or open circuits) or problems with electric connections.
Behavior at startup	At the start of the module the 4-20 mA sensor must be active and able to provide data in line with the set parameters.

ⓘ This kind of sensor will be shown in Gemnis Studio only on the modules equipped with analog input, provided of “C” terminal.

This type of sensor was designed to control an analog input type 4-20 mA.

The configuration of the sensor allows to detect some but not all of the possible failure of the analog sensor.

In particular, this sensor is capable to detect:

- The failure of the device due to short circuits or disconnections;
- unusual currents in the device, below or above the set thresholds.

⚠ This sensor alone is not able to detect:

- The “out of calibration” of the device 4-20 mA, or loss of linearity of conversion or the lock on a fixed value within the range of permissible current.
- Change of the response time of the device
- Mechanical problems, such as the physical detachment of the device from the support that holds it up to the environment to be measured.

 2024 © Pizzato Elettrica S.r.l.

Some of these faults can be excluded by the user, case by case, based on the analysis of the machine or the type of the 4-20 mA device. For example, the exclusion of a fault on the out of calibration of the sensor can be made using sensors that are certified to the level of security required. Otherwise the possibility of mechanical detachment of the sensor can be excluded depending on the quality of its mechanical installation.

This sensor shows the analog output detected by the 4-20 mA device. The user can choose whether to express this value through a BOut byte (8 bits) output expressed in units of 0.1 mA or alternatively through a WOut type Word (16 bits) output expressed in units of 0, 01 mA. For example, a 100 value of BOut indicates that the current in the sensor is 10.0 mA while a 1205 WOut value indicates that the current is 12.05 mA.

The BOut is the default output type and requires fewer resources of the module while the WOut output is preferred when is requested a greater detection accuracy.

ⓘ The BOut output imposes a maximum detectable value, dependent on the capacity of representation of the B type variable, equal to 25.5 mA. Average values higher than this limit will be represented, however, as 25.5 mA.

ⓘ The WOut output type allows, on the contrary, to represent values greater than 25.5 mA, but in this case the user should value in the documentation the range of values for which is guaranteed the accuracyand / or the linearity of conversion.

This software sensor also provides the ability for the user to set a maximum and minimum current such that if the 4-20 mA device exits from the working admitted range is set a mistake. Typically, the thresholds are set slightly outside the normal range (minimum current of 3.8 mA, maximum of 21 mA), but the user can adjust at will according to the device. The exceeding the maximum or minimum threshold sets immediately the error condition.

This sensor allows several types of detected error. For the sake of simplicity, we preferred to have a single output that is activated in case of a fault (XFault) and report on another variable the code of the error (BFaultCode). In this way, the expert user can understand which error occurred and possibly integrate a

routine to manage errors differently depending on the type of the error. We remind that the disappearance of error conditions, their outputs will reset.

The available error codes are:

BFaultCode	Description
0	No error
1	Current sensor below the minimum set
2	Current sensor exceeds the maximum set
100	No power on terminals A or breakage of the internal ADC converters
200	Internal error of the board

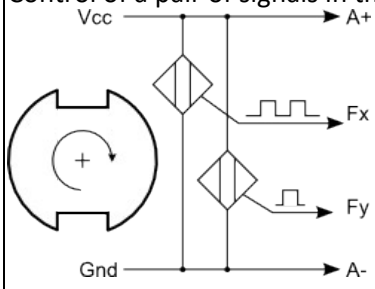
In the case of error code 100 or 200, the read value may not make sense and therefore the output value from the function of the sensor is set at the maximum value allowed by the type of data, or 255 in the case

 2024 © Pizzato Elettrica S.r.l.

of BOut and 65535 in the case of WOut.

The board error is prevalent on other errors. In case of sensor power failure, for example detachment ofthe connector (terminal type) is shown the error 100 and not 1. The basic principle is that an error with higher value is prevalent on those with lower value.

5.7.2.15 SSP2 Sensor

Electrical type	Control of a pair of signals in the frequency range up to 4 kHz.  The power supply of the sensors can be taken in parallel with the module (Vcc = A1, A2 = Gnd) or by separate power supply.
Examples	Speed control and stop motors or other rotating parts. 
Connections	See the examples. With A+ we mean the power supply terminal and with A- the ground terminal of the board to which the external sensors are connected. The nomenclature of A + and A- is dependent on the module (see General Hardware Characteristics).
Parameters	Maximum permissible difference between the values measured by the two channels (consistency) above a minimum threshold Maximum allowable non-coherence time Possibility to activate the detection of the direction of rotation
Outputs	WSpeed: the average value in frequency detected by the two sensors XZero: indicates when both sensors don’t detect movements for 1s. XCw: indicates a clockwise rotation (when possible) XCcw: indicates a rotation in the counterclockwise direction (when possible) XFault: Active in case of error BFaultCode: In case of error indicates the numeric error code
Device failures detectable	Some failures of sensors Inconsistency between the values in the frequency of the two sensors, and then all the possible failures that cause this inconsistency.
Start-up behavior	One or both of the sensors must be active. If you require the detection of the direction of rotation, at the start the engine must be stopped or at least generate on the sensors a frequency below the threshold of detection FRD.

ⓘ This type of sensor is displayed in Gemnis Studio only programming modules with frequency inputs, type "F", suited to the detection of a pair of signals in frequency from 1 Hz to 4 KHz having a predefined shape such that at least one of the two signals is always present.

 2024 © Pizzato Elettrica S.r.l.

Frequency measurement

This type of sensor was made to control redundant frequency inputs, usually representative of the speed of rotation of a motor. The purpose is to detect a frequency signal by a pair of devices, typically a pair of inductive sensors, and calculate the average value as the representative value of the speed of the system. At the same time, we want to verify that the values of the two frequency signals are coherent, or rather that the difference between them is below a set percentage.

Failure consistency in frequency allows to identify faults such as:

- Failure of a single device, for example because short-circuited or disconnected, or oscillation on a frequency inconsistent with that in the measurement.
- Change the response time of the devices (which leads to an inconsistency of signals)
- Some mechanical problems, such as the physical detachment of one of the two devices from the support that connects it to the object to be measured, or a removal of the device from the object to be measured such that there is a "loss of pulse" to the frequency measurement.

ⓘ The module detects and calculates frequency signals by the detection of two pulse sequences, properly shaped, from devices that capture the alternation of these cams on a "phonic wheel" connected to the rotating mechanism that you want to monitor. Please pay attention to the fact that the speed of rotation of the mechanism is not equal to the measured value, but depends on the number of teeth of the phonic wheel. Since normally the speed of rotation of the engine is expressed in revolutions / minute the following conversion ratios are applied:

$$\text{Measured frequency (Hz)} = \frac{\text{Engine rpm} \times \text{N. of teeth of the phonic wheel}}{60}$$

$$\text{Engine rpm} = \frac{\text{Measured frequency (Hz)} \times 60}{\text{N. of teeth of the phonic wheel}}$$

The user can set the maximum allowable percentage difference between the two values F1 and F2, called FD%, and the maximum TF in which this condition is permissible. The difference is calculated with reference to the signal F1. When F2 exits from the range F1±FD% there is an inconsistency. The sensor software will set the output XFault when F1 and F2 are constantly inconsistent for a time greater than TF

(from 0 to 2.54 seconds).

The possibility to set a time inconsistency is very useful to avoid that spurious peaks and the values of F1 or F2 (typically induced noise) bring the system to error condition. However, you should consider that the introduction of a filter time also increases the response time of the module to a fault. The two values therefore must be set in the function of the physical characteristics of the equipment and of the sensors connected to it.

The inconsistency between the two signals is not calculated when the average frequency of the signal is lower than a threshold value Fmin settable by the user. This is necessary to avoid that at low rotation speed the % difference allowed between the two signals is reduced to zero values (expressed in units of 0.1 Hz) or in any case to values that are too low to avoid that the signals are perfectly consistent. For example, setting a maximum difference between the signals of 2% we have that around 10Hz frequencies, the two signals must differ by a maximum of 0.2 Hz to be consistent. To avoid this effect, the user can set a threshold of 0 to 25.5 Hz below which the coherence of the signals is no longer evaluated.

 2024 © Pizzato Elettrica S.r.l.

Motor stopped

This sensor is used to detect frequency values close to 0 Hz that is a measure of "motor stopped". For this reason, the sensor provides an output defined XZero that is activated when the input frequency of the signals of both channels is less than or equal to 1 Hz for at least one second.

The user may not use this output and monitor directly the function WSpeed with a comparator (see operator GEQ) set with a threshold at will, depending on the application.

This sensor can identify certain types of faults on the sensing devices that are important to ensure the correct determination of zero speed. In particular:

- Failure on consistency of activation of the sensors, in particular the case in which both are off. This allows to detect the disconnection of an electric cable or a sensor or (using safe sensors) a fault situation of the device.
- No power or incorrect voltage values on the stage that feeds the devices (A + terminal of the module). This allows you to detect detachments of power supply that belong to both devices.

⚠ This sensor alone is not able to detect common mode failures that affect simultaneously on both sensing devices such as:

- Disconnection of the mechanical support that holds up both devices to the organ in motion or rupture of the phonic wheel.
- Detachment of the power supply of both devices on a point downstream of the terminal A+ that validates the supply voltage of the devices.
- Short circuit of the output stages of both devices at the same time or however in the period of time between two starts of the organ in rotation of the output stages (for example due to strong overvoltages)

For the safe detection of zero speed with this method are therefore very important some precautions that do not relate to the module but the method of construction of the machinery and the wiring of the devices mounted, among which we mention:

- Separate wiring of the devices to minimize the risk of a short circuit between cables, and in particular to avoid point of branching of power supply of the two devices downstream of the terminal A+.
- The choice of various sensing devices (of different type or brand) to minimize the risk of common cause failures.
- Assembling of the sensors on supports in order to avoid the possibility of contemporary detachment of both devices or otherwise or to permit the exclusion of such a failure.

Finally, even if the sensor software should be used only to define the zero speed of an engine is however recommended to make a verification in frequency of the two sensors in order to detect the inconsistency of the two signals and in this way discriminate the case of a single device with output with a short circuit.

Defined F1 and F2 the value of the input frequencies of the two channels this sensor shows in the output their average value expressed in 0.1 Hz units. So, for example a WSpeed value of 1205 indicates that the average frequency is 120.5 Hz.

 2024 © Pizzato Elettrica S.r.l.

Direction of rotation

Using appropriately shaped cams and activating the flag "Enable verification of the direction of rotation" this sensor has two additional logic outputs defined:

XCw indicating a clockwise rotation

XCcw indicating a counter-clockwise rotation

The detection of the direction of rotation is done by identifying the time difference between the phase shifts of the signals of the two sensors and is therefore only possible under certain "mechanical" conditions and only as long as the difference is measured precisely by the board. The general principle provides that the detection of the direction of rotation is made at low speed below a minimum frequency of said FRD (usually equal to 25 Hz, for further details see the technical specifications), and then maintained at high speed. It follows that there are also constraints of maximum acceleration of the engine for determining the direction of rotation.

At very low speeds, next to 0Hz, there may be inconsistencies between the directions of rotation identified by processors, and so it follows that, under FRD is allowed the case where both outputs XCw and XCcw are disabled.

On the contrary it is not allowed that the system is in high speed (frequency of both channels exceeding FRD) with no direction of rotation identified. In this case the module activates the Fault output.

⚠ This sensor, if enabled, is able to detect the direction of rotation of a cam only if certain parametersare met, and in particular:

- The shape of the phonic wheel of detection and the positioning of the device follows strict rules (seetechnical documentation)
- The acceleration of the phonic wheel must be lower than the prescribed maximum values.
- The sensing devices must be able to provide appropriate signals to the requirements, in particular they mustn’t introduce excessive phase delays such as to prevent a correct calculation of the phaseshifts between the signals processed by the module.

 2024 © Pizzato Elettrica S.r.l.

Errors

This sensor allows several types of detectable error. For simplicity we have preferred to have a single output that is activated in the event of a fault (XFault) and report on another variable the code of error (BFaultCode). In this way, the user can go to an expert to understand which error occurred and possibly integrate a routine for error management different from case to case. We remind that the errors are reset at each cycle and then, disappearing the error conditions, they are deleted.

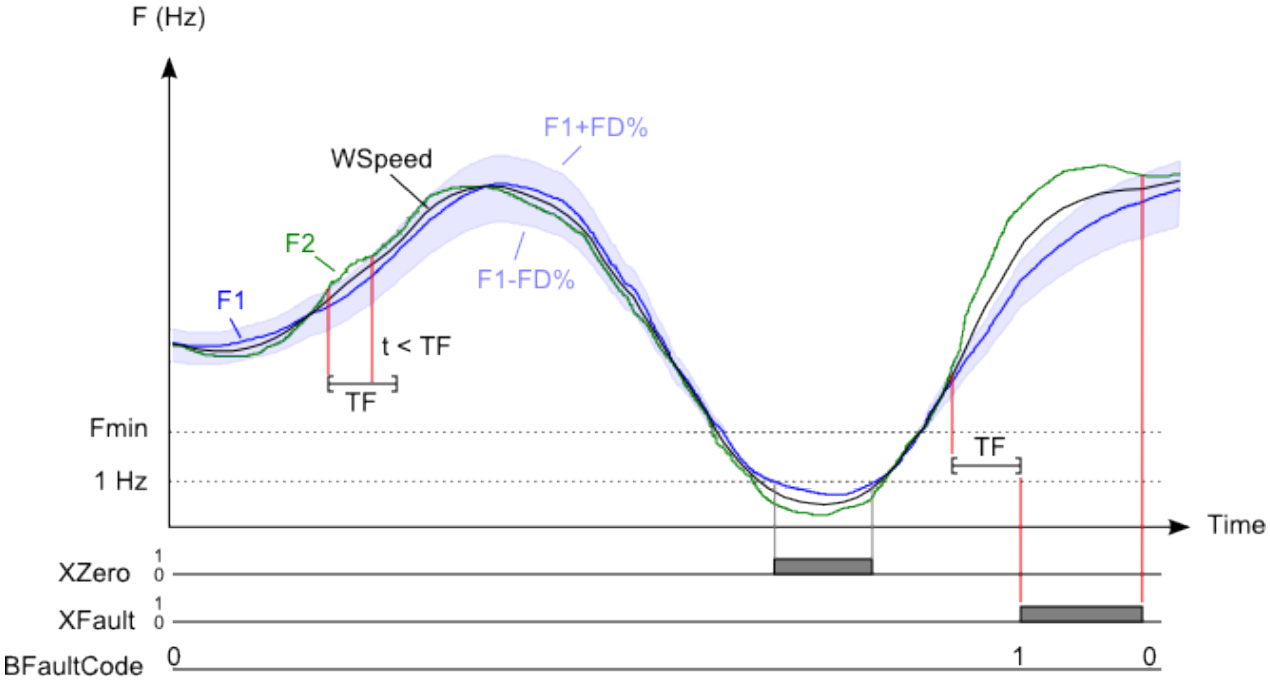
The error codes are:

BFaultCode	Description
0	No error
1	Speed detected by the two sensors too different to the set parameters
10	Both sensors are not present
11	The direction of rotation has been required but the system couldn't give a result, for example due to excessive accelerations or wrong shape of the cams
12	The module has detected input frequencies higher than the maximum allowed (4 kHz).
100	No power on A terminals
200	Internal error of the board

In the case of error code 12, 100 or 200, the read data may not make sense, and then the WSpeed output value is placed on the maximum allowable value of 65535 while the outputs XZero, XCw and XCCw are set to zero.

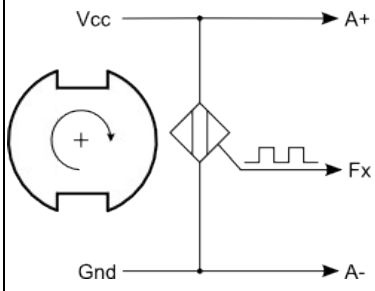
The error of the board is prevalent on other errors. The basic principle is that an error is prevalent with higher value on those with lower value.

In the diagram below, you can see a possible example of the behavior of the sensor as a function of two signals F1 and F2.

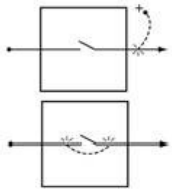


 2024 © Pizzato Elettrica S.r.l.

5.7.2.16 SSP4 Sensor

Electrical type	Control of a frequency signal up to 4 kHz.  The sensor power supply can be taken in parallel with the module (Vcc = A1, A2 = Gnd) or by separate power supply.
Examples	Control speed and stop motors or other rotating parts. 
Connections	See the examples. With A+ we mean the power supply terminal and with A- the ground terminal of the board to which the sensor is connected. The nomenclature of A + and A- is dependent on the module (see General Hardware Characteristics).
Parameters	None
Outputs	WSpeed: the value in the frequency detected by the sensor XZero: indicates when the module does not detect movement for 1 second. XFault: Active in case of error BFaultCode: In case of error indicates the numeric error code
Device failures detectable	Overspeed of the sensor
Start-up behavior	

ⓘ This type of sensor is displayed in Gemnis Studio only on modules with frequency input, type "F", suited to the detection of signals in frequency from 1 Hz to 4 kHz.

This sensor alone is not able to detect electrical faults of the device such as the short circuit to positive or to the mass of the connecting cables or the detachment of the signal cable. It is not even able to detect functional failures such as breakage or short circuit of the output transistor of the device. The output of this sensor should not be used alone for safety related functions.	Possible faults 

This type of sensor is born to control a frequency input up to 4 kHz, usually the speed of rotation of a motor. Since this sensor alone is not able to detect many types of fault, it should not be used alone but always in conjunction with other signals to produce a safety function of functional type.

For example, two or more sensors of this type can be used to check a correlation of movement between two mechanical members which must act jointly. With a first sensor we can monitor the frequency of rotation of a motor and with a second one the frequency of rotation of a second wheel connected to the motor through a pulley. The function performed by the module would then verify the correlation between

 2024 © Pizzato Elettrica S.r.l.

the frequency of rotation of the two bodies and, in this way, that the devices on the field work and that the ratio between the two values is maintained at preset ratios (thus highlighting the problems of slippage or others). A failure of one of the devices would then be intercepted, not so much by the safety function of the sensor (XFault output) but from the Application Program.

ⓘ The module processes signals in frequency by the detection of a sequence of pulses coming from a device that captures the alternation of the cams present on a "phonic wheel" connected to the rotating mechanism that you want to monitor. Please pay attention to the fact that the speed of rotation of the mechanism is not equal to the measured value, but depends on the number of teeth of the phonic wheel. Since normally the speed of rotation of the engine is expressed in revolutions / minute apply the following conversion ratios:

$$\text{Measured frequency (Hz)} = \frac{\text{Engine rpm} \times \text{N. of teeth of the phonic wheel}}{60}$$

$$\text{Engine rpm} = \frac{\text{Measured frequency (Hz)} \times 60}{\text{N. of teeth of the phonic wheel}}$$

This sensor can detect some possible faults on the input devices in particular verify that:

- The frequency of the input signal is lower than the maximum permissible value for the module (4 KHz).

Motor stopped

This sensor can also detect frequency values equal or close to 0 Hz that is a measure of "motor stopped". For this purpose, the sensor provides an output defined XZero that is activated when the input frequency is lower than or equal to 1 Hz for at least one second.

You may not use this output and go to directly monitor the function WSpeed by a comparator (see operator GEQ) set with a desired threshold.

This sensor can detect certain types of faults on the detection device which are important for the detection of zero speed. In particular, we note:

- No power or incorrect voltage values on the stage that feeds the devices (A + terminal of the module). This allows to detect detachments of the power supply of all devices.

⚠ This sensor alone is not capable of detecting the following types of failures:

- Short circuit or opening of the stage of output of the device.
- Short circuit to positive of the cable that give power supply to the device.
- Disconnection of the cable that brings the frequency signal to the module.
- Mechanical disconnection of the support that holds on the device to the moving organ or breakage of the phonic wheel.
- Removal of the power supply on a downstream point of the Ax terminal that validates the supply voltage.

For a safe detection in the zero speed with this method are therefore necessary additional measures, in particular the testing of the signals coming from this sensor with other signals correlated.

 2024 © Pizzato Elettrica S.r.l.

Defined as “F” the value of the input frequency this sensor reports in output their value expressed in units of 0.1 Hz So for example a WSpeed value of 1205 indicates that the average frequency is 120.5 Hz.

This sensor allows several types of error detected. For simplicity, we preferred to have a single output that is activated in the event of a fault (XFault) and return on another variable of the error code (BFaultCode). In this way, the user can go to an expert to understand which error occurred and possibly integrate a routine for error management different from case to case. We remind that the errors are reset at each cycle and then, disappearing the error conditions, they are deleted.

The error codes are:

BFaultCode	Description
0	No error
12	The module has detected input frequencies higher than the maximum allowed (4 kHz).
100	No power supply on A terminals
200	Internal error of the board

In case of error code 12, 100 or 200 the read data may not make sense and therefore the output value from the function of the sensor is placed at the maximum allowed value equal to 65535 on WSpeed while the output XZero is set to zero.

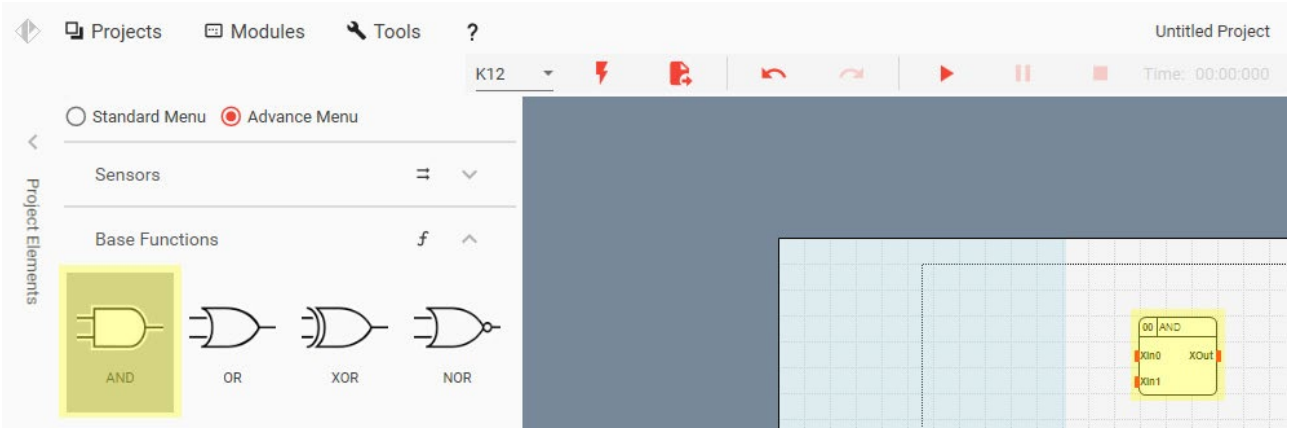
The error on the board is prevalent on other errors. The basic principle is that an error with higher value is prevalent on those with lower value.

 2024 © Pizzato Elettrica S.r.l.

5.7.3 Functional blocks

The Functional Blocks

From functional blocks Tab you can select one Functional Block with the mouse and drag it to the desktop area. Inside Gemnis Studio this is equivalent to creating a function whose input variables and output are available on "connection points" that can be connected to other points within the desktop.



Each functional block so created has on the left side connection points related to the input of the function and to his right-side connection points related to output variables. The coded name of the safety function is written in the upper part of the box while on the upper left is automatically created a progressive ID useful to discriminate more functional blocks of the same type.

Within the area of the functional block are represented the names of input and output variables, close to the respective connection point. In the lower part of the functional block are sometimes present frames that contain the currently selected values for the parameters of the function.

The function associated to each block can have several input and output connection points and can produce may output data, whose meaning and values depend block-by-block. It is necessary that you read the documentation associated with each block (see later) to fully understand them.

Variables of functions can be of different types (see box) the variable name and the color of the connection point are helpful to understand the type.

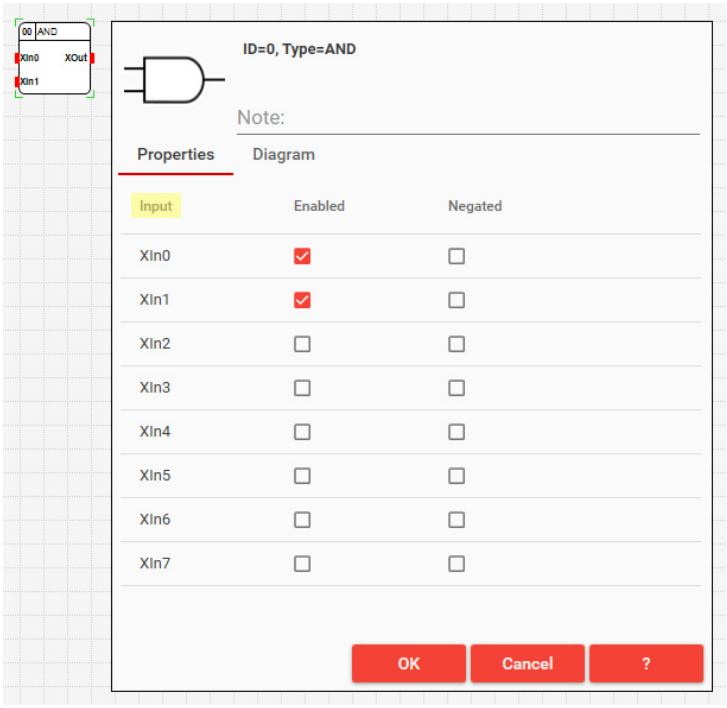
Unlike the sensors, the connection points of the functional blocks must always be all connected otherwise the program will not compile.

> ⓘ Studio Gemnis is able to manage binary and numeric type information. The binary information (0,1) are identified by the prefix "X", the numerical 8 bit information by prefix "B" and the 16 bit information by prefix “W”.

Changes on functional blocks

Nearly every functional block can be modified and to do it just clicks with right mouse button on the functional block area, and then click the block button Property to access the Edit tab (or simply double-click it) or click on the button Delete to delete it from your desktop. Each block has its own specific properties but some features are common. Moreover, each functional block is associated with a specific point of this document, which you can immediately access by the help button of the form (the icon with the “i” symbol).

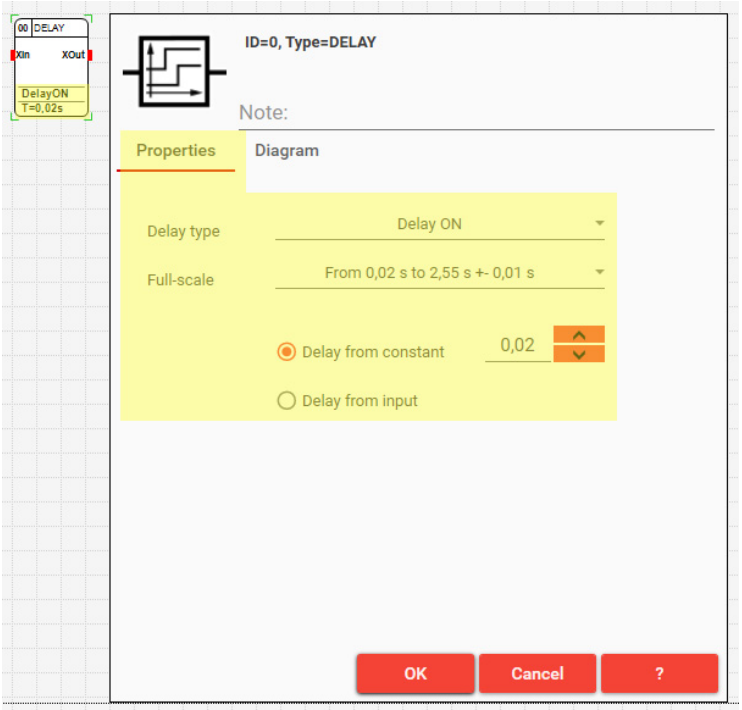
 2024 © Pizzato Elettrica S.r.l.



Parameters

Almost all functional blocks are both configurable by values and by variables. It is possible to change the underlying function for a functional block not only on the basis of some constants, but also vary the number of inputs or outputs. For example, the functional blocks related to basic logical functions (AND, OR, etc.) allow a number of inputs which vary from 2 to 5 while other functional blocks such as DELAY allow the modification of the type and value of the delay times.

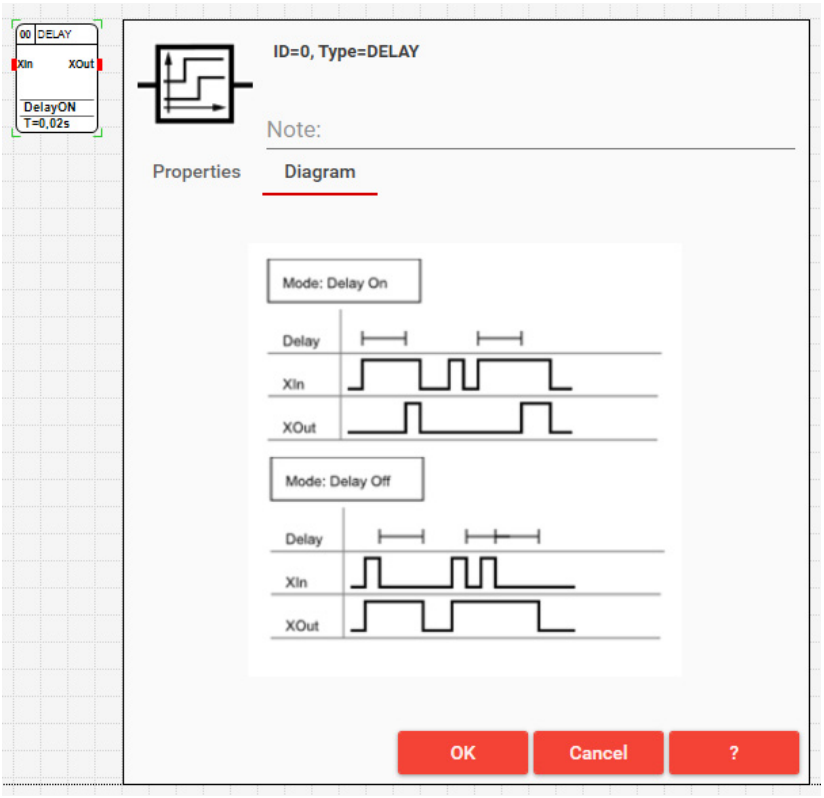
The parameters of a functional block are listed in their own tab and you can modify them. The parameters that are not immediately visible (such as the number of inputs or outputs) and theirs value are graphically displayed in small rectangles that inside the functional block.



 2024 © Pizzato Elettrica S.r.l.

Diagram

In the properties of each functional block is present the Diagram Tab that shows a summary of the function carried out by functional block. For more information, please click on the info icon to be routed to the definition of functional block within this document.



 2024 © Pizzato Elettrica S.r.l.

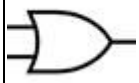
5.7.3.1 Functional block AND

Name	AND	Base Boolean function AND 
Inputs	XIn (n)	Inputs of type X. Each input can be individually negated.
Parameters	Inputs number	From 2 to 8.
Outputs	XOut	Logical AND function of the inputs

The logical AND function output is equal to 1 only when all inputs (eventually after the possible negation) are 1. The following example shows the truth table of the AND function with two inputs (XIn0 and XIn1).

XIn0	XIn1	XOut
0	0	0
0	1	0
1	0	0
1	1	1

5.7.3.2 Functional block OR

Name	OR	Base Boolean function OR 
Inputs	XIn (n)	Inputs of type X. Each input can be individually negated.
Parameters	Inputs number	From 2 to 8.
Outputs	XOut	Logical OR function of the inputs

The logical OR function output is equal to 1 if at least one input (eventually after the possible negation) is 1. The following example shows the truth table of the OR function with two inputs (XIn0 and XIn1).

XIn0	XIn1	XOut
0	0	0
0	1	1
1	0	1
1	1	1

 2024 © Pizzato Elettrica S.r.l.

5.7.3.3 Functional block XOR

Name	XOR	Base Boolean function XOR 
Inputs	XIn (n)	Inputs of type X. Each input can be individually negated.
Parameters	Inputs number	From 2 to 8.
Outputs	XOut	Logical XOR function of the inputs

The logical XOR function output is equal to 1 if an odd number of inputs (eventually after the possible negation) is 1. The following example shows the truth table of the XOR function with two inputs (XIn0 and XIn1).

XIn0	XIn1	XOut
0	0	0
0	1	1
1	0	1
1	1	0

5.7.3.4 Functional block NAND

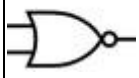
Name	NAND	Base Boolean function NAND 
Inputs	XIn (n)	Inputs of type X. Each input can be individually negated.
Parameters	Inputs number	From 2 to 8.
Outputs	XOut	Logical NAND function of the inputs

The logical NAND function output is equal to 0 only when all inputs (eventually after the possible negation) are 1. The following example shows the truth table of the NAND function with two inputs (XIn0 and XIn1). The NAND function is the logical negation of function AND.

XIn0	XIn1	XOut
0	0	1
0	1	1
1	0	1
1	1	0

 2024 © Pizzato Elettrica S.r.l.

5.7.3.5 Functional block NOR

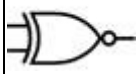
Name	NOR	Base Boolean function NOR 
Inputs	XIn (n)	Inputs of type X. Each input can be individually negated.
Parameters	Inputs number	From 2 to 8.
Outputs	XOut	Logical NOR function of the inputs

The logical NOR function output is equal to 0 if at least one entry (eventually after the possible negation) is

1. The following example shows the truth table of the NOR function with two inputs (XIn0 and XIn1). The function NOR is the logical negation of function OR.

XIn0	XIn1	XOut
0	0	1
0	1	0
1	0	0
1	1	0

5.7.3.6 Functional block NXOR

Name	NXOR	Base Boolean function NXOR 
Inputs	XIn (n)	Inputs of type X. Each input can be individually negated.
Parameters	Inputs number	From 2 to 8.
Outputs	XOut	Logical NXOR function of the inputs

The logical NXOR function output is equal to 0 if an odd number of inputs (eventually after the possible negation) is 1. The following example shows the truth table of the NXOR function with two inputs (XIn0 and XIn1).

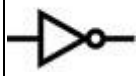
NXOR function is the logical negation of function XOR.

Note that the NXOR function is equivalent to the "equal" function, i.e. XOut output is active when XIn0 equals XIn1.

XIn0	XIn1	XOut
0	0	1
0	1	0
1	0	0
1	1	1

 2024 © Pizzato Elettrica S.r.l.

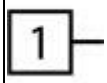
5.7.3.7 Functional block NOT

Name	NOT	Base Boolean function NOT 
Inputs	XIn	
Parameters	None	
Outputs	XOut	Logical NOT function of the input

The logical function NOT inverts the XIn input. Below is the table of truth about NOT function.

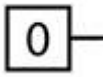
XIn	XOut
0	1
1	0

5.7.3.8 Functional block TRUE

Name	TRUE	Base Boolean function TRUE 
Inputs	None	
Parameters	None	
Outputs	XOut	Fixed output value of 1.

The functional block TRUE output is always 1. This function is useful as input to other functional blocks in case you need to set an always active value.

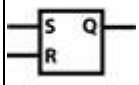
5.7.3.9 Functional block FALSE

Name	FALSE	Base Boolean function FALSE 
Inputs	None	
Parameters	None	
Outputs	XOut	Fixed output value of 0.

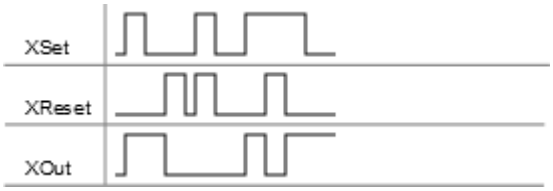
The functional block FALSE output is always 0. This function is useful as input to other functional blocks in case you need to set an always off value.

 2024 © Pizzato Elettrica S.r.l.

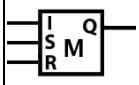
5.7.3.10 Functional block SET/RESET

Name	SET/RESET	Basic memory logic function, Reset dominant 
Inputs	XSet XReset	When input XSet is active sets output to 1 When input XReset is active sets output to 0
Parameters	None	
Outputs	XOut	Value of output. At startup value is 0.

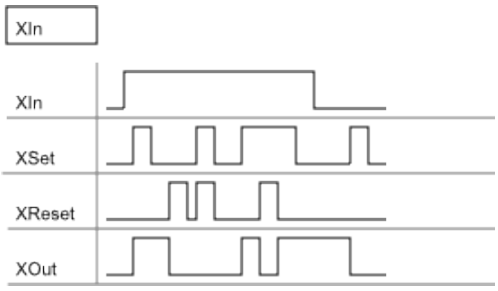
The functional block SET/RESET is the basic logic memory function. When XSet input becomes equal to 1 XOut output is set to 1 and remains so even if XSet back to 0. When XReset input becomes equal to 1 XOut output goes to 0 and remains so even if XReset back to 0. In case of contemporary activation of XSet and XReset the latter is prevailing and XOut output goes to 0 (Reset dominant).



5.7.3.11 Functional block MEM

Name	MEM	Generic memory function, Reset dominant 
Inputs	XIn or BIn or Win XSet XReset	The Input with the value to store When input XSet is active sets output to Input value When input XReset is active sets output to 0
Parameters	None	
Outputs	XOut or BOut or WOut	Value of output. At startup value is 0.

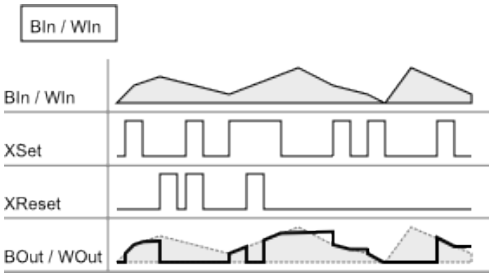
The functional block MEM is the generic memory function and can store the value of the input variable activating the input XSet.



Unlike the functional block SET/RESET this block allows you to store data of type X, B and W. Whatever the type of input data, when input XReset is activated the output becomes equal to

0. In case of simultaneous activation of the inputs XSet and XReset, is the latter that prevails and the output is forced to 0 (reset dominant).

In the case of X type variables pays attention that activating XSet stores the value assumed at input XIn, and then, if this were to 0, the output is set to 0.



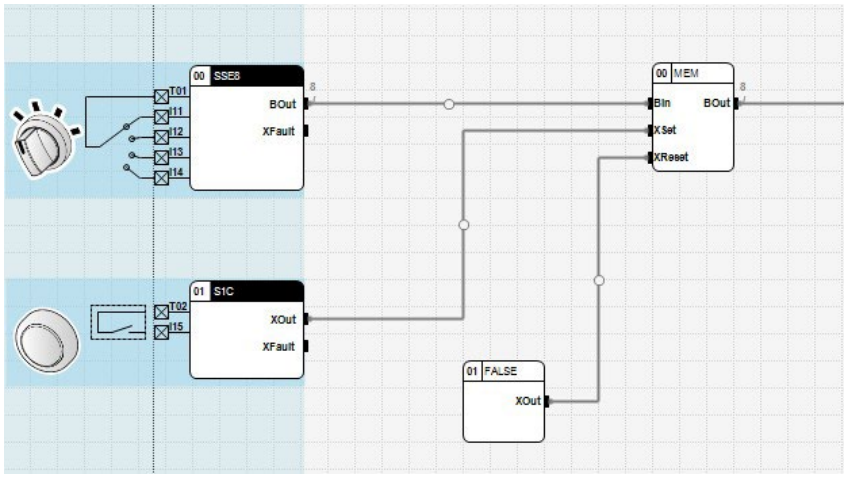
The basic functional block SET/RESET can be interpreted as a specific case of this functional block with the input of type X always set at 1.

This block is usually used for storing working conditions and to avoid that they can be changed involuntarily, or to allow the change only to those who have the necessary authorizations. Consider the case where a modal selector is used to vary the functions performed by a

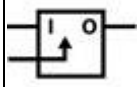
 2024 © Pizzato Elettrica S.r.l.

machine. If you want to change the settings should be confirmed by means of a button you can use the MEM functional block (see example in the picture) by connecting in cascade with Selector sensor, and using the button as confirmation of storage. If you want only some users are allowed to change these settings you can use a key-enabling contact instead of the button.

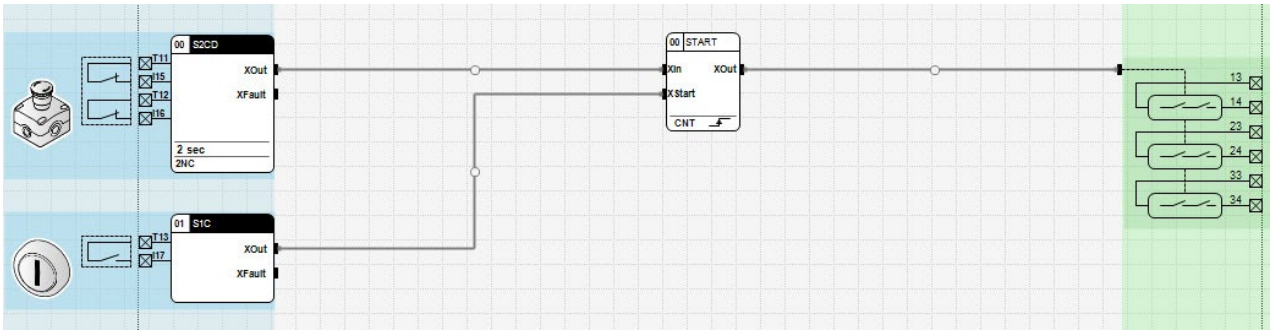
Finally, pay attention to the fact that at power-on the MEM function has output value equal to 0. In the example in picture even if the switch indicates the start position 1 the output of the MEM function will remain at 0 until the button is pressed.



5.7.3.12 Functional block START

Name	START	Control function for start process 
Inputs	XIn XStart	Is the input signal to be controlled Is the controller signal
Parameters	Start Type	Can take three values: Controlled Start on XStart rising edge (default) Controlled Start on XStart falling edge Manual/Automatic Start, activated by the presence of XStart
Outputs	XOut	Output value

The functional block START is used to control an XIn input signal. When XIn input has value 1 that value is transferred on output as soon as the XStart input satisfies the condition of the parameter. As soon as the XIn input takes the value 0 is immediately transferred on XOut output. This function is used for the processes activation functions and to prevent automatic restarts.

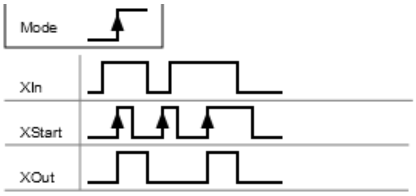


 2024 © Pizzato Elettrica S.r.l.

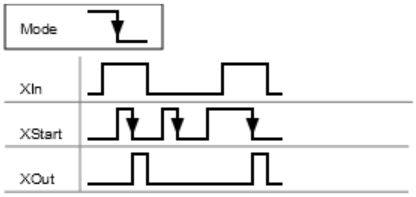
Through this function it is possible to check out some external failures, such as sticking of the Start pushbutton contacts.

The functional block START can take three different configurations:

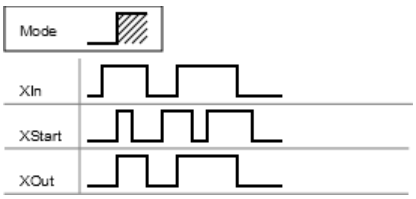
1. Start checked on rising edge of control signal XStart (default configuration). In this configuration, the output signal XOut is activated (if the XIn input is 1) when you press the Start button.



2. Start checked on the falling edge of control signal XStart. In this configuration, the output signal XOut is activated (if the XIn input is 1) when you release the Start button. This configuration is useful when the same Start pushbutton makes another task on rising edge. You can perform two different actions in sequence with a single button.



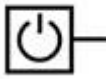
3. Start active on the presence of control signal XStart. This configuration does not allow control over the Start pushbutton but it is very useful to verify the feedbacks of external contactors connected to the outputs of the module.



⚠ The Start function active on the presence of the start control signal (version 3) can also be used as startup function but must be verified that, if the Start pushbutton sticking, machine cannot automatically restart unexpectedly. To prevent this, other functional blocks may be necessary, or eventually to check that machinery operation mode do not allow such an event.

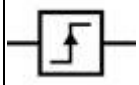
 2024 © Pizzato Elettrica S.r.l.

5.7.3.13 Functional block POWER ON

Name	POWER ON	Signal active only in the first program execution cycle 
Inputs	None	
Parameters	None	
Outputs	XOut	The first cycle is 1, then 0 is always

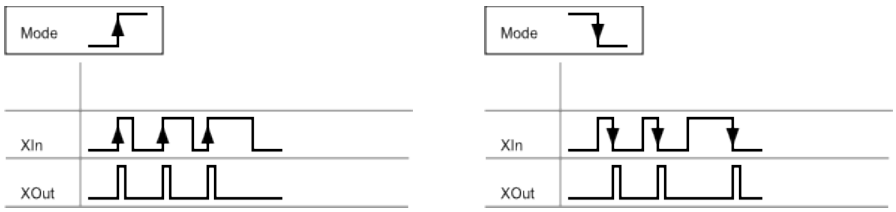
This functional block is useful to eventually make some tests at every restart of the module. For instance, this block can force execution of a test on control devices of machinery each time you start it.

5.7.3.14 Functional block TRIGGER

Name	TRIGGER	Detect the rising or falling edge of an input signal 
Inputs	XIn	Input to be evaluate
Parameters	Type face	The chosen edge (rising or falling) of the input signal.
Outputs	XOut	Is active only at detection of the chosen edge of XIn

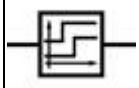
The TRIGGER functional block evaluates a XIn input and generates in XOut a pulse of duration equal to one cycle of execution when it detects the chosen front of the signal XIn.

If the function provides the detection of the rising edge and at the start XIn is positive, on XOut a pulse is generated in output.

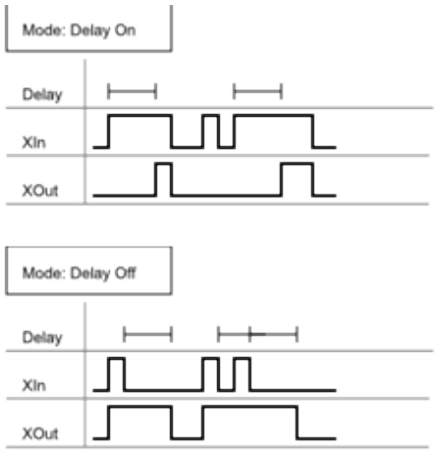


 2024 © Pizzato Elettrica S.r.l.

5.7.3.15 Functional block DELAY

Name	DELAY	Returns a type Delay Off or Delay On signal 
Inputs	XIn (BTime)	The input signals (Optional) the value of time Delay
Parameters	Delay Type Full Scale Time or input	Whether to perform a Delay On or Delay Off function Indicates the full scale value of the delay Indicates whether the delay will be fixed or variable according to the value of a new input BTime or WTime
Outputs	XOut	The delayed signal

This block makes Delay On or Delay Off type timings.



A “Delay On” delay type implies that XOut activates later than XIn with a delay equal to the set time. As soon as XIn goes to 0 also XOut immediately deactivates. If during the initial timed phase, the input signal goes to zero and then back to 1 the timer restarts.

A “Delay Off” delay type implies that XOut activates immediately when XIn is active but, when XIn is deactivated, XOut continues to remain active, with a delay equal to the set time. If during the late timed phase, the input signal is set and reset again, the timer restarts from the moment of the last resetting.

The delay time can be expressed as a fixed value or a variable.

In the case of fixed delay, it is necessary to select the full scale of the timer that you intend to use, and then specify the time within the permissible range. There are several ranges and, function of the chosen one, the resources used and the precision of timing change. In particular ranges are:

Code	Full Scale	Unit of measure	Precision	Resource Type/Input
1	from 0.01 to 2.55 s	0.01 s	-0 +0.01 s	B
2	from 0.1 to 25.5 s	0.1 s	-0 +0.1 s	B
3	from 1 to 255 s (4 min)	1 s	-0 +1 s	B
4	from 1 to 65535 s (18 h)	1 s	-0 +1 s	W

Note so that it is possible, with the same timing, have very different precisions. For example, you can set a delay time of 0.8 seconds using either the full-scale Code 1 or Code 2 but that the former has a better precision.

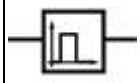
If you choose to use a variable time as a function of an input, then in the functional block a new input appears, having the same type of resource type used. So, if for example you chose the full-scale code 1 the new input will have “BTime” name, it will be of B type, and values in the input connection point will be interpreted in 0.01 seconds units of measure. If instead we chose the full-scale code 4 then the input would name “WTime”, it will be a W type data, and values present in this input will be interpreted in 1 second units of measure. Changing the type of full scale can therefore changes the type of the connection point. In this case, any existing connections will be deleted given the diversity of two types of underlying data (Gemnis Studio does not allow connections between different data types).

If the input BTime (or WTime) change its value while the timing process is in execution, the new value will be compared, on each execution cycle, to time already passed. So, if BTime increased its value, there will be

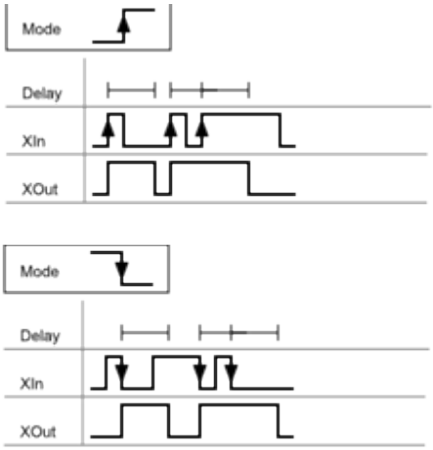
 2024 © Pizzato Elettrica S.r.l.

an automatic extension of the Delay duration. In the case BTime decrease its value up to descend below the time already passed there would be the immediate Delay termination.

5.7.3.16 Functional block PULSE

Name	PULSE	Returns a Delay Off signal type selected on the edge of the input 
Inputs	XIn (BTime)	The input signal (Optional) the value of time Delay
Parameters	Edge type Time or input	The chosen (rising or falling) edge of the input signal. Indicates whether the pulse duration will be constant or variable according to the value of a new input BTime
Outputs	XOut	The signal generated from XIn depending on the parameters selected

This block generates a fixed length signal on XOut starting counting the time on chosen edge of XIn input signal.



If the chosen edge is the rising one, the XOut output signal will immediately activate with XIn and will stay for the chosen time. If during the timed phase, the input signal deactivates and activate again, the time count restarts from the moment of the last rising edge of XIn.

If the chosen edge is the falling one, XOut activate when XIn goes from on to off and will stay for the chosen time. If during the timed phase, the input signal is activated and deactivated again, the time count restarts from the moment of the last falling edge of XIn.

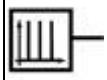
The delay time can be expressed as a fixed value or a variable.

In the case of fixed delay is sufficient to give the time (from 0.01 to 2.55 s) that you intend to use.

If you choose to use a variable time as a function of an input value then in the functional block appears a new input named “BTime” and values present in this entry will be interpreted in 0.01 seconds units of measure. So, for example a value of 15 in the BTime input is interpreted as a time of 0.15 seconds. If the input BTime change its value while the timing process is in execution, the new value will be compared, on each execution cycle, to time already passed. So, if BTime increased its value, there will be an automatic extension of the pulse duration. In the case BTime decrease its value up to descend below the time already passed there would be the immediate pulse termination.

 2024 © Pizzato Elettrica S.r.l.

5.7.3.17 Functional block CLOCK

Name	CLOCK	Generates pulses with pre-established fixed rate 
Inputs	None	
Parameters	Clock Type (Period)	The clock type (Optional) the clock period
Outputs	XOut	The clock output

This functional block generates an infinite sequence of pulses having 10 ms duration and frequency depending on the chosen Clock and in particular:

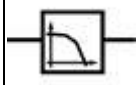
1. Fixed Clock: generates a pulse every 0.1 seconds
2. Fixed Clock: generates a pulse every second
3. Fixed Clock: generates a pulse every hour
4. Adjustable Clock: generates a pulse according to the selected time. The period is selectable between 0.01 and 2.55 seconds with a precision of -0 +0.01 seconds.
5. Adjustable Clock: generates a pulse according to the selected time. The period is selectable between 0.1 and 25.5 seconds with a precision of -0 +0.1 seconds.
6. Adjustable Clock: generates a pulse depending on the period. The period is selectable between 1 and 255 seconds with a precision of -0 +1 second.
7. Adjustable Clock: generates a pulse according to the selected time. The period is selectable between 1 and 65535 seconds with a precision of -0 +1 seconds.

The adjustable clock uses more system resources.

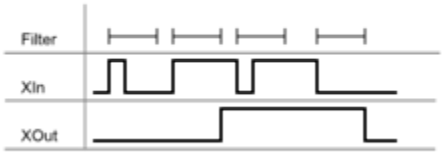
Please note that the internal clock of the module is not suitable for measurements of time with the precision required to play day clock function.

 2024 © Pizzato Elettrica S.r.l.

5.7.3.18 Functional block FILTER

Name	FILTER	Filter noise from a signal 
Inputs	XIn (BTime)	Incoming data to evaluate (optional) Input filtering time (by 0.01 to 2.55 sec)
Parameters	Time or input	Indicates whether the filtering time will be fixed or variable according to the value of a new input BTime
Outputs	XOut	The filtered output. At startup it is 0.

The functional block FILTER, as the name indicate, filters XIn input signal from certain noise, and return the filtered output signal XOut. The filtering function is made so that, whatever the input signal, the output signal will not change until the input signal does not remain constant for the minimum set time.



This functional block can be very useful in the presence of incoming signals strongly disturbed or to eliminate any contacts rebounds with excessive duration.

Note that the Gemnis series modules filter in hardware contacts rebounds having a less than 10 ms duration. This filtering alone is sufficient for the vast majority of mechanical devices.

⚠ Attention! Inserting this software filter introduces a delay in detecting the input signal which you must consider in assessing the overall response time of the module.

 2024 © Pizzato Elettrica S.r.l.

5.7.3.19 Functional block ERROR

Name	ERROR	Set the module in Error state (see ERROR State.) 
Inputs	XIn	Input that, when activated, puts the module in Error state
Parameters	Error code	The module error code (from 1 to 15).
Outputs	No	

This functional block has a function different from everyone else because, if invoked, sets the module in Error state with a particular error level, which consists in:

1. Block execution of the Application Program in the moment when this functional block is invoked
2. Put the module in safe state with all safety outputs open
3. Keep unsafe output signal in the last state achieved
4. Highlight the error code on LED of processors (P1 and P2) using a sequence of flashes which indicate the Error Code.
5. Send the same error code on communication ports, including USB port
6. Continue to generate pulsed signals on T type clamps in order to maintain the active view of the LED associated with inputs.

This functional block is generally used to stop execution of the program when the Application Program has detected a malfunctioning external device such that gravity is necessary to block entirely, in safe mode, the functions of the module.

Almost all Sensors report an output called “XFault” that, if active, implies the detection of a malfunction on the external device connected, such as a short circuit. Typically, this type of malfunction does not allow continuation of safety functions performed by the module but it requires the intervention of a maintainer who verify the failure, resolve it and finally restart machine. The functional block ERROR is typically connected to the Fault output of various Sensors in order to highlight, with different error codes, where the fault occurred. The parameter "error code" allows you to indicate, with a number from 1 to 16, a code that will be displayed as a sequence of blue flashing on the processor LEDs and that will be send on the communication ports (USB, COM, for other ports see the documentation attached) with a string such as "[ERROR xx]" where xx is the Error Code.

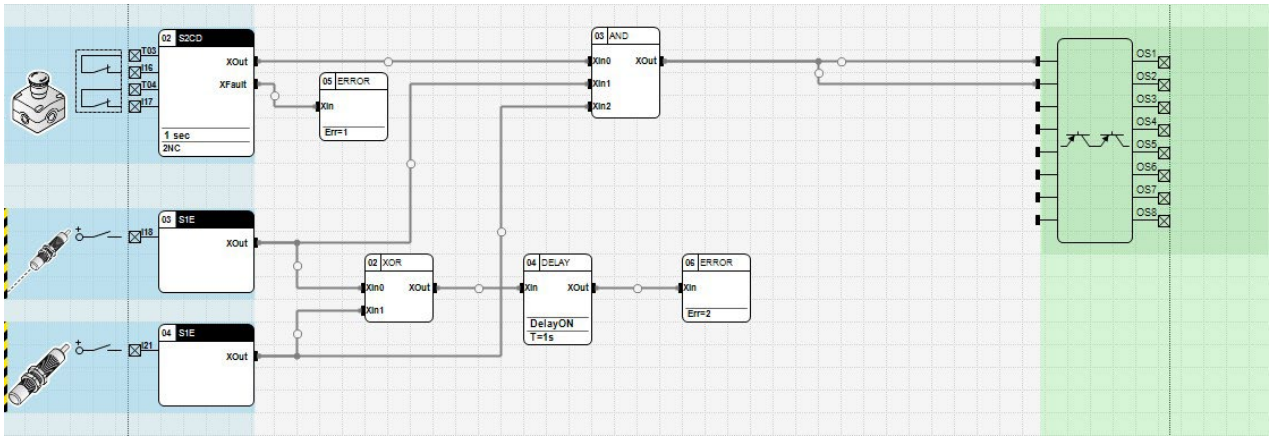
The functional block ERROR can also be invoked in other parts of the Application Program, such when using many unsafe Sensors configured to detect functional failures anyway. For example, in the case of a set of devices where a given situation can't physically happen, detecting that condition can still be conveyed on a functional block ERROR for detection of a fault.

This structure allows a very flexible fault management because it is possible, for each Sensor or in any State analysis, to choose different behaviors in case of fault detection. Programmer can opt between an immediate stop, a managed stop, specific error code for the most important cases, a common error code across multiple devices (putting in OR the respective XFault signals) and so on.

Note, as example, diagram below where we assume that a machine has an emergency pushbutton for emergency stops and two inductive sensors, both connected to the same guard, and that therefore should activate and de-activate always almost simultaneously.

(Note: without additional measure this example is not safe, it is only for ease of analysis).

 2024 © Pizzato Elettrica S.r.l.



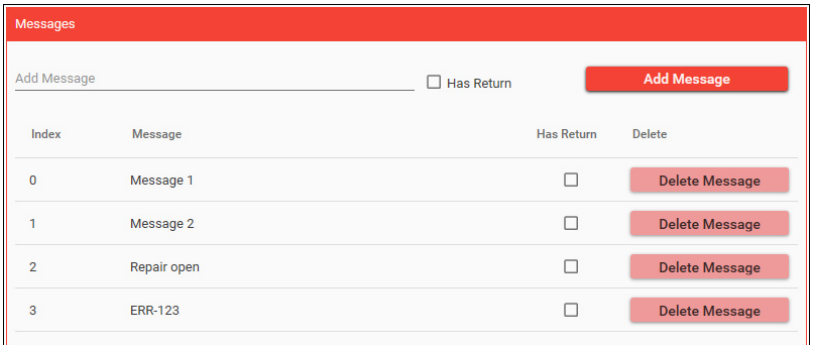
In this example, the three Sensors are AND connected so that in case one of the three devices opens the emergency exit opens. If the Sensor that controls the mushroom detects a fault, the module stops with error code 1. The two Sensors that control the inductive sensors devices are not able to detect external failures but knowing that they must move coherently we can detect at least a functional failure in case one of the two has a failure or a short circuit. The XOR circuit, downstream of the two S1E type Sensors output, turns on when the two signals are different. To avoid problems due to small rebounds the XOR functional block output is connected to a timer DELAY ON type with a fixed time of one second. Only if the two sensors have different outputs for a time greater than 1 second, then the DELAY functional block activate the ERROR ones, which will put the module in Error state with error code 2.

 2024 © Pizzato Elettrica S.r.l.

5.7.3.20 Functional block MESSAGE

Name	MESSAGE	Sends a message on port USB and COM 
Inputs	XIn (BMsg)	On the rising edge of XIn appends a string on the transmission buffer (Optional) Entry with the ID of the message to be sent
Parameters	Fixed message or from input Send Timer	Indicates whether the message will be chosen from the list messages or if it will be the numeric value input BMsg Indicates whether the selected message will be preceded by the value of the internal timer (only from Kernel 11)
Outputs	No	

The functional block MESSAGE sends a message on the USB port. Messages to be sent must have been previously loaded into the Messages Tab within the current Project (see 5.7.1). Messages loaded are automatically indexed with a progressive ID, 0-based. In the picture you can see an example where 4 messages are loaded.



To load a message just type a text in the Message field, and then press the button Add Message. The flag “Has Return" indicates whether after the text you want to add a newline character.

A maximum of 127 messages can be inserted and they can be up to 127 characters long one but it is recommended, to avoid unnecessarily wasting of program memory and to not consume transmission bandwidth, to keep them the shorter possible. You can use all non-control ASCII characters (0x20 to 0x7F from) with the exclusion of the square brackets and braces that are reserved. All messages generated independently by the module are in fact enclosed in square brackets or braces, such as "[ERROR 01]", so that any external program listening can discriminate the strings generated by the application program and those generated by the module itself.

Sending the message occurs on the positive edge of XIn, to avoid the generation of multiple and consecutive messages.

Alternatively, you can send a variable message, that is the numerical value (expressed in hexadecimal) of the Code input, during the rising edge of input XIn.

If you are using a module with Kernel 11 (or greater) is possible by setting the appropriate flag, precede the above message identified by the value of the internal timer when sending a message. This feature can be useful to identify the time when the event occurred.

The internal timer expresses the value in seconds (4 bytes) and centiseconds (1 byte) passed from the time the module. Such values are expressed in hexadecimal.

If you are using a module with Kernel 11 is possible, by setting the appropriate flag, precede the above identified message by the value of the internal timer when sending a message. This feature can be useful to identify after the time when the event occurred.

 2024 © Pizzato Elettrica S.r.l.

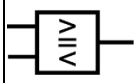
The internal timer expresses the value in seconds (4 bytes) and centiseconds (1 byte) passed from the start of the module. Such values are expressed in hexadecimal.

> ⚠ Please note that messages should not perform safety functions and serve solely for informational or debugging functions. In case multiple messages are generated simultaneously from the Application Program, they will be inserted, in the order of generation, inside the transmission buffer and then transmitted in sequence. If there is an excessive generation of messages, such as to fill the transmission buffer, the last messages to be transmitted may be lost (one reason more to create the shorter possible strings).

Messages are sent simultaneously from the USB port.

> ⚠ USB port, by default, is enable to send messages generates by the Application Program. However, if they are generated in excessive amounts, can be stacked to monitor and debug commands that are normally transmitted between Gemnis Studio and the module and slow down or prevent the display of the module status. To disable the transmission of messages on USB port it is necessary to set the control flag in Project form (only for modules with Kernel 11), or use the System Command P06.

5.7.3.21 Functional block EQU/GEQ/LEQ

Name	GEQ/EQU/LEQ	Make a numerical comparison between two values of type B or W and indicates the result in Boolean format (X) 
Inputs	BIn or WIn (BCode or WCode)	Data value to be compared (Optional) Second input with the data value to compare
Parameters	Data type: B W Comparison type: GEQ EQU LEQ Compare with: Constant Input XCode	 Performs a comparison between B type (Bytes) value Performs a comparison between W type (Word) values Set XOut if XIn > = reference value Set XOut if XIn = reference value Set XOut if XIn < = reference value The reference value is a constant The reference value is the second Input, BCode or WCode
Outputs	XOut	Indicates whether the result of the comparison is true or not

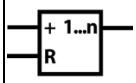
The functional block GEQ (EQU, LEQ) makes a comparison between Byte or Word data type. The comparison varies depending on the " Comparison type" parameter which can be:

- GEQ (Greater or EQual). Checks if the input is greater or equal to the reference value.
- EQU (EQUal). Checks if the input is equal to the reference value.
- LEQ (Lesser or EQual). Checks if the input is lesser or equal to the reference value.

The input value can be compared with a constant or with a second input, in each case of the same numerical type. Comparisons between different types of data are not allowed.

 2024 © Pizzato Elettrica S.r.l.

5.7.3.22 Functional block COUNTER

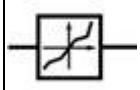
Name	COUNTER	Pulse counter 
Inputs	XIn XReset	Input data to be counted Counter reset input
Parameters	Limit Value XIn Edge	The maximum value of the counter, once reached the counter restart from 0 Indicates whether the count is made on rising or falling edge of XIn
Outputs	BOut	Counter value. At startup it is 0.

This function counts the number of edges (rising or falling) present on input XIn and reports the count on BOut. When the Limit Value is exceeded, the counter is reset to 0. When XReset input is activated, counter BOut will be reset.

The maximum value allowed for the Limit Value is 254.

The function is XReset dominant, so if XIn and XReset are simultaneously activated BOut is set to 0.

5.7.3.23 Functional block LKTBL

Name	LKTBL	Conversion table of data of the same type 
Inputs	BIn	Incoming data to convert
Parameters	Data Number	The number of data table
Outputs	BOut	Converted data output. At startup it is 0.

The functional block LKTBL (LOOKUP TABLE) converts an input value of type B in another output value of type B through a two-column conversion table having data chosen by programmer. In practice the input value of BIn is compared with the list of n values ("Data Number" parameter) inside the first column of the table. If the data is found then BOut takes the value in the second column of the same table otherwise BOut is set to 0.

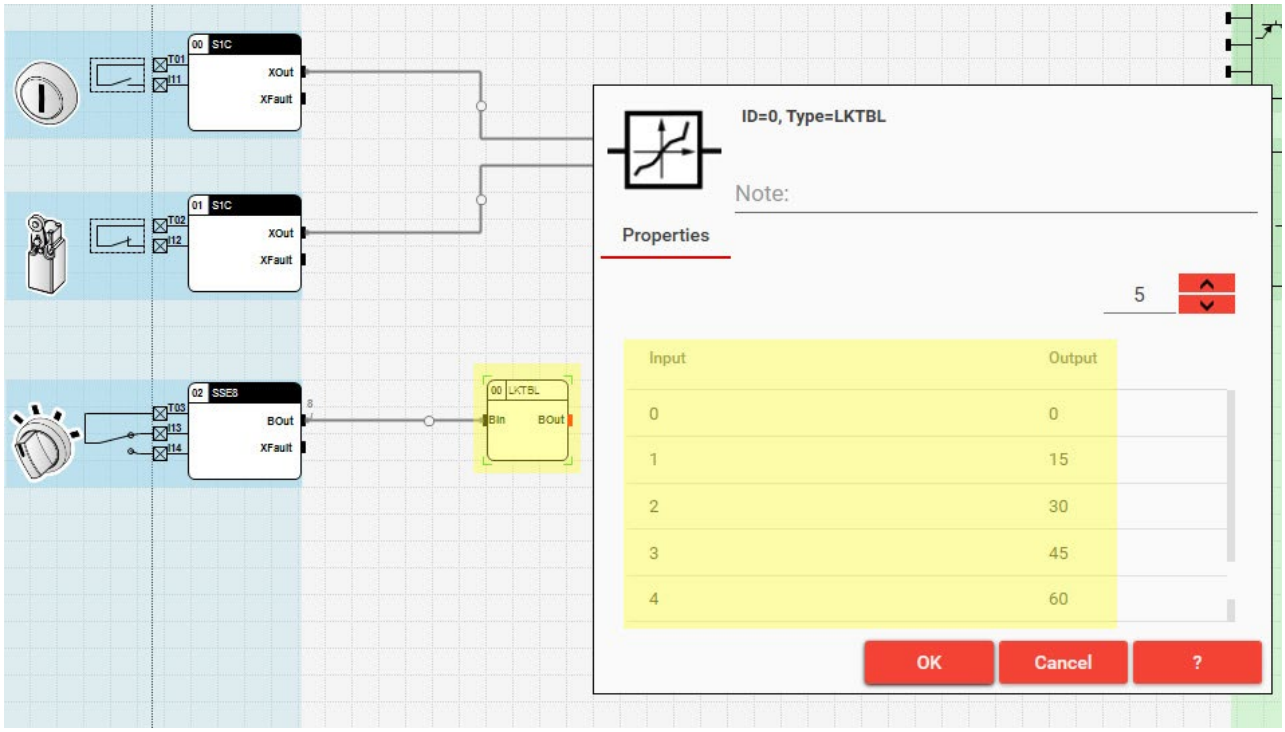
This type of block is used in the conversion of data from data sources of type B in order to adapt them to other blocks.

In the following example, a machine is equipped with an emergency stop and a magnetic sensor which control a guard. The safety output OS1 stops immediately the machine as soon as the guard will open or if emergency mushroom is pressed. A second safety output OS2 is controlled by a DELAY function (delay off)

where delay time is variable depending on the position of a modal switch. Data from the selector have values ranging from 1 to 4 (0 in case of failure) as a function of selector position. To convert these values in different time values it is used a LKTBL functional block having a 5 values table. As you can see in the figure, the selector switch in the first position generates a 1 value that is converted from LKBL block 15 value which is in turn interpreted by the DELAY function as 1.5 sec value (15 expressed in units of 0.1 seconds means 15 × 0.1 = 1.5 seconds) of the delay time. Similarly, in positions 2, 3 and 4 values are converted into time delay of 3, 4.5 and 6 seconds. Note that a data in the table was also provided for the case in which the LKTBL input is 0, i.e. when the selector sensor has identified a fault and then produces an output value of 0, in this case setting that the second safety output stops the delay immediately having given a data equal to 0.

(Note: this example lacks totally the sensors fault management).

 2024 © Pizzato Elettrica S.r.l.



Warning: If the input value is not present in the table for default LKTBL output has the value 0.

5.7.3.24 Functional block LDC (LOCK DOOR CONTROL)

Name	LDC	Functional block to control door lock system 
Inputs	XClosed XLocked XLock XUnlock XReset	Input that indicates the door is closed Input that indicates the door is locked Input to request the door locking Input to request the door unlocking Input to reset the functional block in case of error
Parameters	P0: Locking Mode P1: Lock type P2: Unlock type P3: Delay XOut P4: Unlock autonomous P5: Error filter	Indicates whether the door is locked with XDL output off (default, spring locking principle or "D") or active (locking principle “E”) Indicates the evaluation mode of the XLock signal Indicates the evaluation mode of the XUnlock signal Allows to specify a short delay on the XOut activation Allows the system to unlock in an autonomous way, without control signals Sets a filter time for the error conditions.
Outputs	XOut XDL XFault BState	Output indicating that the door is closed and locked Door lock Signal Output indicating the detection of a fault condition The state of the functional block

This functional block has been specifically made to simplify development of Application Programs which need to control the lockable guards of machinery. It takes account of various combinations which may happen in the real world and typical problems due to the mechanical movement of the physical devices.

The inputs have the following functions:

- XClosed: The signal from the device that detects the position of the door. Can come from a

 2024 © Pizzato Elettrica S.r.l.

separate device or from the same device that provides for the locking of the door. It is understood that the input signal is positive when the door is closed.

- XLocked: The signal from the device that will block the port (i.e. switch with solenoid). A positive value indicates that the door is locked.
- XLock: The input signal to the functional block which ask to block the door (where feasible). This input can be evaluated on the edge of the signal or on its presence. Typically, if the signal comes from a control button this input is detected on the edge of the signal, while if it comes from a supervision function is evaluated on the presence of the signal.
- XUnlock: The input signal to the functional block which ask to unlock the door (where feasible). This input can be evaluated on the edge or the presence of the signal. Typically, if the signal comes from a control button this input is detected on the edge of the signal, while if it comes from a supervision function is evaluated on the presence of the signal.
- XReset: If the functional block identifies a fault it goes in an error state, from which it is possible to exit only by activating this input, or by rebooting the module.

The parameters allow the following mode:

Parameter P0

The locking of the door is controlled from the XDL output. There are two mode of locking:

- Principle D. This is the default principle and implies that, without further action, closing the door locks it automatically through an internal mechanism (typically spring, "locked by spring") and it is unlocked by turning on the XDL output that typically feeds a solenoid that releases the locking spring. This principle is considered safer in case of machinery where the danger continues even after shutting down the machine (because of parts with inertia, or high temperature) since a power failure does not allow for the immediate opening of the door.
- Principle E. Implies that the door closing normally remains unlocked and to lock it is necessary to activate XDL output which typically provides to feed an electromagnet which locks the door. This principle is frequently used to avoid wrong operations in machines such as the opening of a door in the middle of a production cycle.

Parameter P1

The lock request has three different modes:

- Request for block on the rising edge of XLock. It is the default mode and provides that, at the detection of the rising edge of signal XLock, the system goes in state 21 and there will remain for the maximum time indicated in the parameter. If within this time XLocked becomes positive state is set to 30 otherwise returns to 20.
- Request for block on the falling edge of XLock. It is identical to the previous case with the only difference that the start of the process occurs on the negative edge of XLock.
- Request for block evaluated on the presence of the signal XLock. In this case, as long as this signal remains active, the functional block remains pending confirmation of locking of the door and, when XLocked becomes positive, goes to state 30, otherwise to the release of XLock, returns to state 20.

Parameter P2

The unlock request has three different modes:

- Request for unlock on the rising edge of XUnlock. It is the default mode and provides that, at the detection of the rising edge of signal XUnlock the system goes in state 31 and there will remain for the maximum time indicated in the parameter. If within this time XLocked goes to 0 the state is set to 20 otherwise returns to 30.

 2024 © Pizzato Elettrica S.r.l.

- Request for unlock on the falling edge of XUnlock. It is identical to the previous case with the only difference that the start of the process occurs on the negative edge of XUnlock.
- Request for unlock evaluated on the presence of signal XUnlock. In this case, as long as this signal remains active, the functional block remains pending confirmation of unlocking of the door and, when XLocked goes to 0, the state is set to 20, otherwise, to the release of XUnlock, returns to state 30.

Parameter P3

XOut becomes positive in the state 30 but, if necessary, its activation may be slightly delayed due to parameter P3. It may happen, especially for the very large shelters, that during their closing phase the sensing devices are subjected to shaking such as to create rebounds on the electrical contacts. The time adjustable via the parameter P3 will allow to set a small delay, from 0 to 2.55 seconds, to allow the stabilization of the contacts before considering the guard closed.

Parameter P4

Generally, the locking devices of the door can be unlocked only under the control of the supervision electric system, and not autonomously. This is the default condition, with the parameter P4 disabled. In some cases, however, the machine manufacturer may provide that, specialized operators having access keys, can unlock the protection locally (and in doing so open the contact which detects the locking of the door) causing the immediate stop of the machine or a part thereof. If this happens the LDC block detects a fault condition (state change was unexpected) and goes into error state. If you want to admit the autonomous unlock of the device, you must activate the P4

parameter that allows the direct passage from state 30 to state 20, controlled by the contact XLocked. Pay attention to the fact that, once it goes back in state 20 the locking system is immediately unlocked. To reactivate XOut it will not be sufficient to close the door, but it will be necessary to execute the locking function.

Parameter P5

The functional block LDC detects some fault states, in particular that:

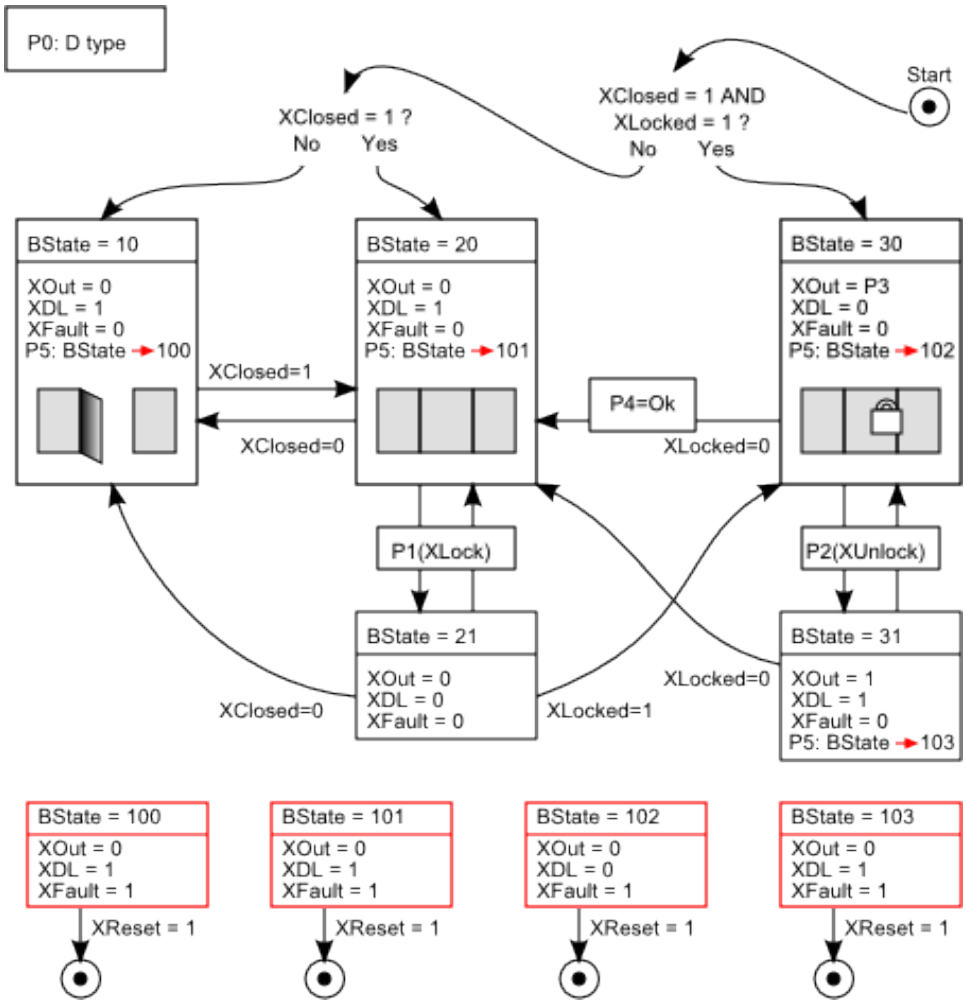
- In the state 10, with open door, the door is not even locked.
- In the state 20, with the door closed, the door is not even locked.
- In the states 30 and 31, with the door closed and locked, the door is not open.

If any of these conditions is detected, the functional block goes in its relative error state.

To avoid that rebounds on the electrical contacts or other small issues lead erroneously the function in an error state, it is possible to set a minimum time, between 0 and 2.55 seconds, for which the error condition should remain, before entering in error state.

Below to the functional block LDC there is the following state machine, valid for the D principle. In the case of the E principle and the state machine is identical, only with inverted XDL output.

 2024 © Pizzato Elettrica S.r.l.



States of LDC functional block and their meaning:

Start: This is the start state of the function. It checks the physical condition of the door. If the door is closed and locked the variable BState is set to 30. If the door is only closed then only is set to 20 to 10 otherwise.

BState 10: It foreseen that the door is open and unlocked. In this state the XOut output of the function is turned off and the XDL output is active (in the case of the principle D) or not active (in the case of the principle E) for keeping the door unlocked, also when it is closed, so as to allow access to the machine.

BState 20: It foreseen that the door is closed but not locked. The output XOut of the function is turned off and the XDL output is active (in the case of the principle D) or not active (in the case of the principle E) for keeping the door unlocked to allow access to the machine.

BState 21: This state happens when there is a request to lock the door and the door is closed but not yet locked. The output XOut is kept turned off but the output XDL reverses its value for trying of locking the door. The mode of entry and exit from this state depend on the parameter P1, "Block type".

BState 30: It foreseen that the door is closed and locked. The output XOut is activated with a delay equal to the parameter P3, delay often useful to wait for the stabilization of the signals (contacts rebounds) coming from the door. The XDL output keeps the door locked.

BState 31: This state happens after a request to unlock the door. During this phase XOut is active and XDL reverses its value trying to unlock the door. The mode of entry and exit from this state depend on the parameter P2, "Unlock type." If the unlock request is enabled on the edge of XUnlock

 2024 © Pizzato Elettrica S.r.l.

signal, the functional block tries to unlock the door for to the time set. If in this time XLocked input goes to 0 the state is set to 20 otherwise returns to 30. In the case instead the unlock request is evaluated on the presence of the XUnlock signal then, as long as this signal remains active, the functional block remains waiting confirmation of unlocking of the door and, when XLocked goes to 0, moves to state 20, otherwise to the release of XUnlock, returns to state 30.

There are then four different error states with code 100 to 103. In all these states, the XOut output is set to zero and the XFault output is set to 1. The different error codes are useful during debugging, in order to understand in what condition, the fault occurred.

BState 100: Error that occurs in the state BState 10 (XClosed = 0 and XLocked = 1) if the door remains locked longer than the time set in parameter P5.

BState 101: Error that occurs in the state BState 20 (XClosed = 1 and XLocked = 0) if the door remains locked longer than the time set in parameter P5; it differs from the BState 100 error, because in the state BState 20 the door is closed but not locked.

BState 102: Error that occurs in the state BState 30 (XClosed = 1 and XLocked = 1) if the door remains open or unlocked longer than the time set in parameter P3.

BState 103: Error that occurs in the state BState 31 (XClosed = 0 and XLocked = 1), after that a request has been made to unlock the protection, if the door is open and locked longer than the time set in parameter P5. It indicates an anomaly in the protection unlocking mechanism.

The performance of XDL changes according to the state in which the error occurred.

To exit these states, it is necessary to activate the XReset input or restart the module. In any case, the functional block starts again from the Start state

Outputs of the functional block.

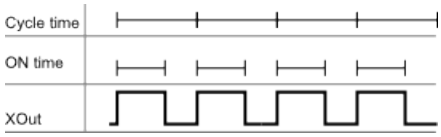
- XOut is the output which, when active, indicates that the door is closed and locked. It is 0 in all other cases.
- XDL is the control output of the locking system. Typically, it is connected directly to an output of the module to drive the external system.
- XFault is active when the block has identified a fault condition of the door.
- BState indicates numerically the status of the machine as described above. This output is useful not only for debugging but also for signaling to the user that requests access to the machine. Using this variable, it is easy to create signal like traffic lights according to the state of the machine.

 2024 © Pizzato Elettrica S.r.l.

5.7.3.25 Functional Block WAVE

Name	WAVE	Generates a waveform with variable period and ON time 
Inputs	none	
Parameters	Cycle time On time	Wave period Time, in the period, where the signal is ON
Outputs	XOut	The output wave

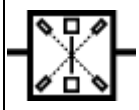
This functional block generates an infinite and constant sequence of digital waves with a cycle time can be set between 0.02 and 2.55 seconds and an ON time can be set from 0.01 to period.



The main use of this functional block is to generate pulsed signals useful for flashing visualization of signals or LED. For example, by putting in AND an error signal with a signal generated by WAVE you will have that, when active, the error signal is flashing.

Note that the internal clock of the module is not suitable to generate waves with the precision necessary to perform a function of a clock.

5.7.3.26 Functional Block MUTE2

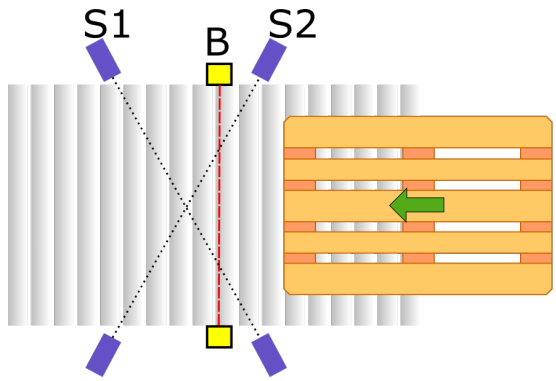
Name	MUTE2	Functional block for the control of a Muting system with 2 beams 
Inputs	XDoor XMute1 XMute2 XReset	The signal from the access control device (barrier). The signal from the first muting device. The signal from the second muting device. The signal that resets the function in case of errors or other
Parameters	TCont TMax TBlind	The maximum contemporaneity time allowed between XMute1 and XMute2. The maximum total time of muting If specified, admits a Blind Time
Outputs	XOut XMute XFault BFault (optional)	Control output Indicates the status of the muting function Indicates whether the functional block has identified a state of Fault Indicates the error code in case of fault

This function allows the bypass (or muting) of a safe device which governs access (optical barrier, mechanical devices, etc..) by signals from two inputs (optical sensors, switches) so-called muting.

The muting function is activated by a particular sequence (temporal logic) inputs. The sequence is linked to more user-customizable parameters.

The typical application of a muting function provides the control of a gap as in the example that follows.

 2024 © Pizzato Elettrica S.r.l.



B = Barrier

S1 = First muting sensor

S2 = Second muting sensor.

Note that, since the input to the object in transit, the point of intersection of the rays of the two sensors is placed behind the barrier to prevent the bypass of the system by a person who is confined to move to the center of the path.

For more information, see the reference regulation (e.g. EN 61496-1:2005).

The Muting function is activated only in response to a safe state of the inputs, ie after XDoor active while XMute1 and XMute2 are off. On the module startup is required that the state of inputs is in a safe condition before you can activate the muting function.

The inputs of the block have the following functions:

- XDoor: The signal from the access control system (light curtain, device connected to the shelter). It is active when the access is not busy (barrier free, shelter closed) or whatever it is in a safe condition.
- XMute1: The signal from the first muting device. It is active when the sensor is occupied (interrupted photocell, operated switch).
- XMute2: The signal from the second muting device. It is active when the sensor is occupied (interrupted photocell, operated switch).
- XReset: The signal that resets the function in case of errors or other. This signal is only evaluated when the functional block is in an error state. It is possible to choose whether the reset occurs on the rising edge or on the falling edge of the signal by activating the options in the Properties window of the MUTE2 block.

The parameters of the function have the following meaning:

- TCont: Indicates the maximum contemporaneity time allowed between XMute1 and XMute2. It can be set from 0.5 to 4 sec. If the sequence of activation of XMute1 and XMute2 exceeds this time the functional block crashes (XFault becomes active) with error code 101 (on output BFault).
- TMax: Indicates the maximum total time of muting. Its range is from 5 to 255 sec. If the muting function remains active for a total time greater than this parameter, the functional block crashes (XFault becomes active) with error code 102 (on output BFault).
- TBlind: Indicates the Blind Time or the time when the Muting function remains active for a short period after XMute1 or XMute2 were released. It is used to filter out spurious signals of the muting sensors due to protruding parts or crevices present in the object in transit, such that the muting function would terminate before sending the required functional block in error. It can be set from 0 to 2.55 seconds. Please note that, when used, this filter extends the muting time of the system. We recommend that you keep this parameter to the minimum value dictated by the knowledge of the machinery.

Outputs of the functional block:

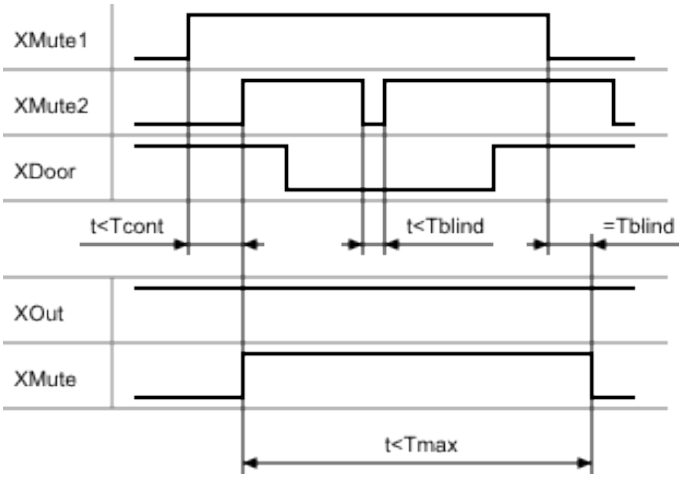
- XOut: It's the control output of the function. When the muting function is not active XOut reply the value of XDoor. When the muting function is active XOut is active regardless of the signal XDoor, or bypasses it.
- XMute: Indicates the status of the Muting function. It's active when the muting function is active

 2024 © Pizzato Elettrica S.r.l.

and vice versa.

- XFault: Indicates whether the functional block has detected a fault condition, such as exceeding the maximum time provided for the muting function.
- BFault: This output is optional and can be disabled. If this is activated, in the case of Fault indicates the error code, otherwise it is always 0.

The following diagram shows the typical sequence of input and output signals of a functional block that performs a correct action of muting. Note that the spurious signal on XMute2 is filtered because of length less than TBlind and contextually the Muting function ends as soon as one of the two Muting signals is restored plus the Blind Time.



5.7.3.27 Functional Block EDM

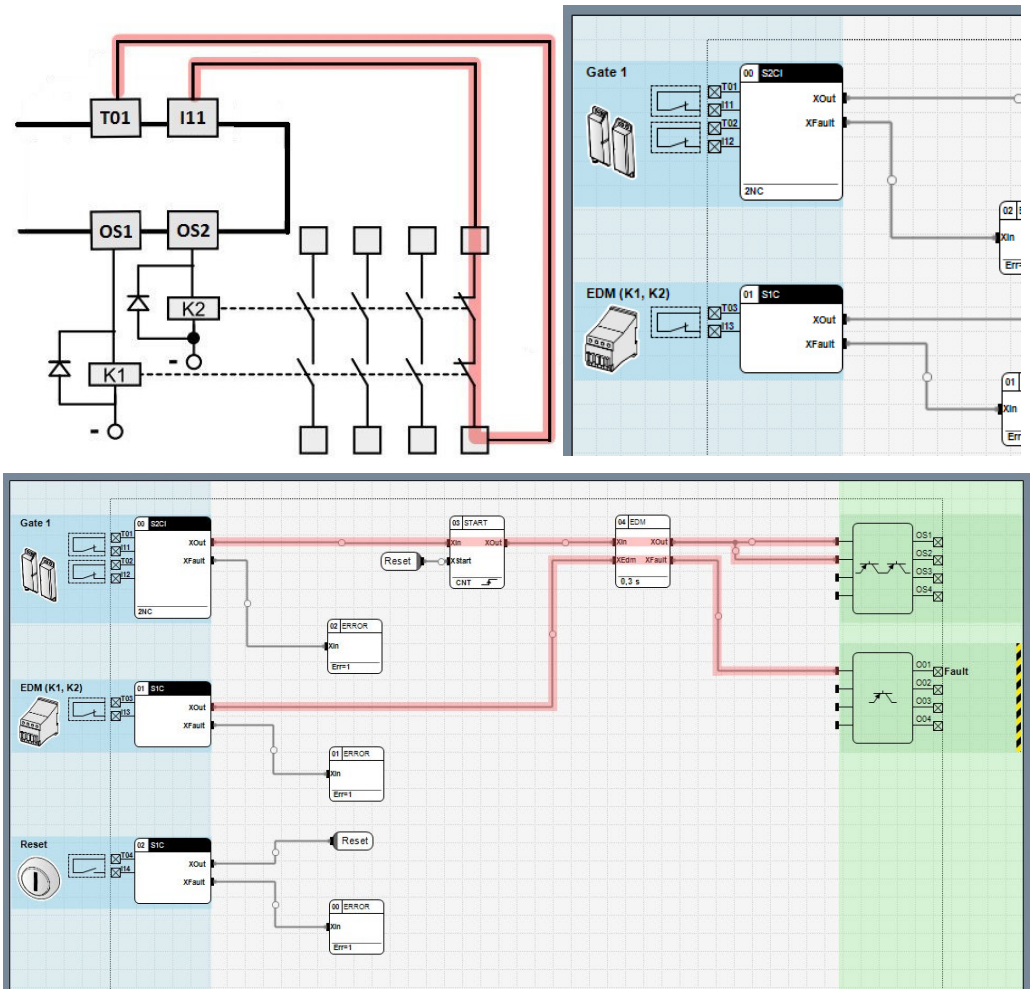
Name	EDM	Functional block for the control of an external device 
Inputs	XIn XEdm XReset	Enable signal of the EDM block Feedback signal from the external device (EDM) The signal that resets the function in case of error
Parameters	TCont	The maximum contemporaneity time allowed between XIn e XEdm
Outputs	XOut XFault	Output signal (activation of external device) Signal indicating the detection of a fault condition

The function EDM (external device monitoring) allows you to check the integrity of the external devices through direct control of their state. The external devices are typically relays or contactors with guided contacts and equipped with at least a normally closed contact to realize the circuit of EDM.

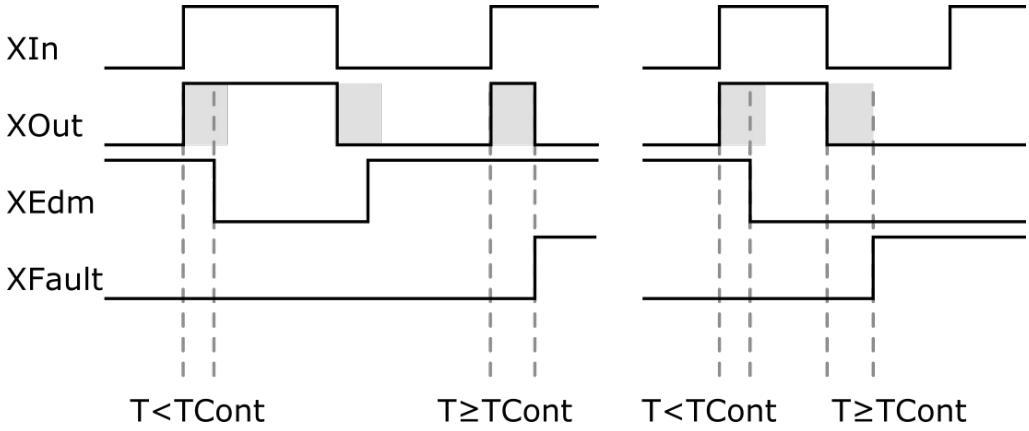
In the case in which one of the external devices is fault, the functional block EDM detects the abnormality by comparing the input XEdm and the input XIn.

The following image shows an application example where the circuit EDM checks the integrity of the two contactors K1 and K2 through the series of their normally closed contacts. The EDM circuit is at the terminals T01 and I11:

 2024 © Pizzato Elettrica S.r.l.



The logic of the functional block checks that the activation of XOut will lead to the deactivation of XEdm, with a maximum contemporaneity time equal to TCon (adjustable from 0.01 s to 2.55 s). In the same way, the functional block verifies that the deactivation of XOut will lead to the activation of XEdm, with a maximum contemporaneity time equal to TCon.



The inputs of the block have the following functions:

- XIn: enable signal of the safe outputs; means active when the external device must be activated (enabled).
- XEdm: feedback signal from external device; this signal represents the state of the external device connected to the safe outputs, and in case of relays or contactors, the circuit is realized by the series of normally closed contacts of the devices.

 2024 © Pizzato Elettrica S.r.l.

- XReset: signal which controls the reset of the functional block in case of error. This signal is evaluated only when the functional block is in error state. It is possible to choose whether the reset occurs on the rising edge or on the falling edge of the signal, by activating the "Show reset input" option present in the Properties window of the EDM block.

The parameters of the function have the following meaning:

- TCont: Indicates the maximum contemporaneity time allowed between XIn and XEdm. It is adjustable from 0,01 to 25,5 sec. If the activation sequence of XIn and XEdm exceeds this time the functional block goes in error (XFault becomes active).

Outputs of the functional block:

- XOut: It is the control output of the function. The output is active when the input XIn is active. It remains active if within the time TCON occurs a deactivation of XEdm.
- XFault: It indicates whether the functional block has identified any condition of Fault, for example for the overcoming of the maximum time established for control of the EDM.

5.7.3.28 Functional Block SERIAL

Name	SERIAL	Functional block responsible for the dialogue between the safety module and the external PLC in order to monitor the status of sensors, logic units or inputs in general that are connected to the module 
Inputs	XStart XIn(n)	Type X input indicating the start of data transmission to the module Type X input indicating the status of an external input
Parameters	Number of bits to send Bit duration Number of inter-character bits IDLE state Type of transmission	It can vary from 2 to 32 (1 bit = 1 state of an external input) It is the time duration of the single bit (it can vary from 10 to 500 ms in synchronous transmission, from 20 to 500 ms in asynchronous transmission) It is the number of stop bits inserted in the data transmission sequence at the end of the bits to be transmitted (it can vary from 2 to 10) IDLE status of the XOut data transmission cable (default status equal to 0, it is equal to 1 when the "Active status IDLE" box is flagged) It can be synchronous or asynchronous
Outputs	XOut XClock	Data transmission output Output used to synchronize the emitter and receiver (for synchronous transmission only)

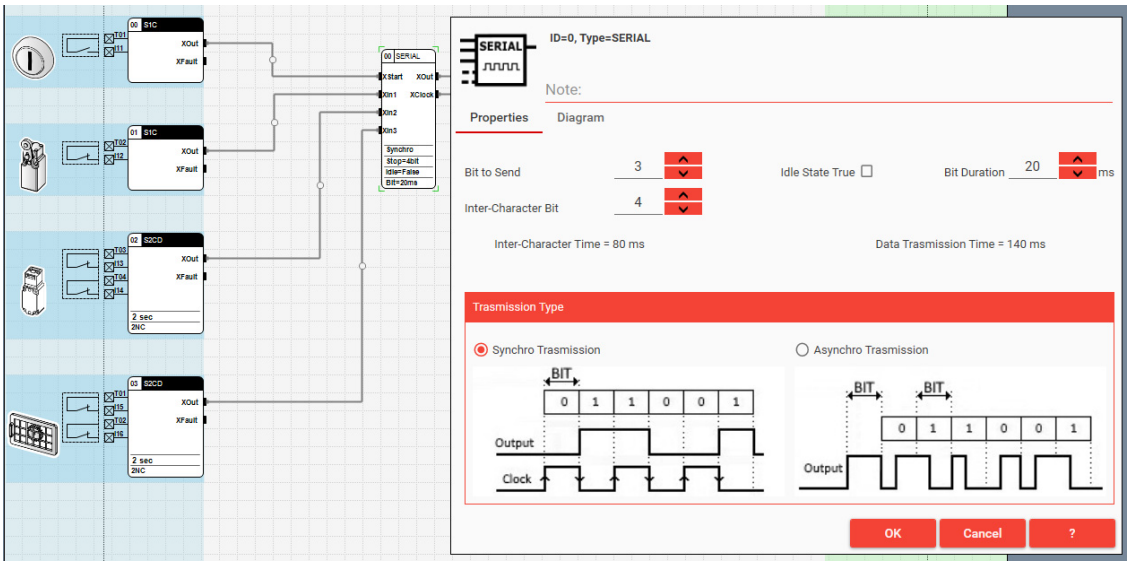
This functional block allows the safety module to send to an external PLC the status of external devices or sensors, as well as the results of logical combinations coming from units of type AND, OR, GEQ, LDC, MUTE etc., which are connected to the module, until a maximum of 32 inputs, using only 1 cable in the asynchronous transmission and 2 cables in the synchronous transmission.

This type of communication allows considerable savings in the wiring, as well as in the number of inputs and outputs used in the PLC.

 2024 © Pizzato Elettrica S.r.l.

Through the synchronous transmission, it is possible to reach a data transmission speed equal to 100 bits per second, through the asynchronous transmission the maximum transmission speed is equal to 50 bits per second, however the advantage of this configuration is that only one cable is used.

The following image shows an example of connection of a SERIAL functional block with 3 inputs, and the corresponding Properties screen in which it is possible to set the data transmission parameters.



Synchronous transmission

In the synchronous transmission, inputs have the following functions:

- XStart indicates when to start the data transmission: when it is 0, the transmission is deactivated, when it is 1 the transmission sequence starts;
- XIn is the status of given data to be transmitted (0 or 1).

When XStart = 0, the functional block does not transmit data. When XStart = 1, the system performs a status sampling of the various inputs, composes the corresponding bit sequence and starts transmitting it through the XOut output. As long as XStart = 1, the inputs sampling is executed before each transmission start of a sequence of bits, so that the functional block is able to detect any status changes of the inputs that occurred during the transmission.

The data transmission takes place through the XOut output. The transmission frequency depends on the set bit duration, which can be between 10 and 500 ms. The bit duration must be chosen according to the program processing speed by the PLC: for each data transmission cycle, it is advisable that the PLC has the time necessary to receive and decode the information.

The XClock output has the function of synchronizing the emitter (safety module) and receiver (PLC); it is in state 0 when the data transmission is turned off.

When XStart = 1, XClock begins to transmit an alternating series of bits (1, 0, 1, 0, etc.) having a time equal to the bit duration previously set.

When the number of the bits transmitted by XClock is equal to the set number of bits to be transmitted (from 2 to 32, equal to the number of sensors to be monitored) a number of inter-character bits is inserted in the transmission, equal to the value set in the Properties screen, indicating to the receiver the end of a data transmission sequence and the beginning of the next sequence.

The IDLE status of the transmission cable is normally off and is 0.

In this state, if the data transmission is deactivated (XStart = 0), the value 0 will be read on the XOut output. When the IDLE status is set equal to 1, even if the data transmission is deactivated, the value returned by

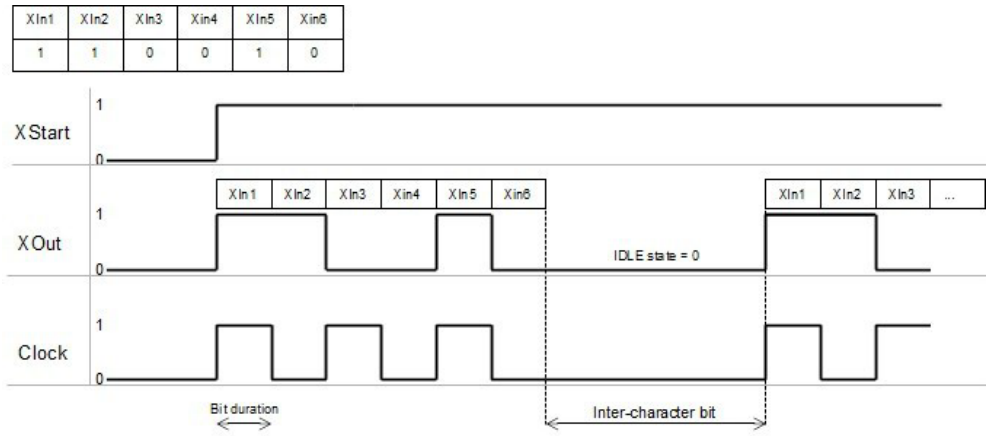
 2024 © Pizzato Elettrica S.r.l.

XOut will be equal to 1: this setting is useful in case the receiver performs monitoring functions on the emitter therefore, in case of any interruption of the electrical connection, the receiver is able to signal the anomaly.

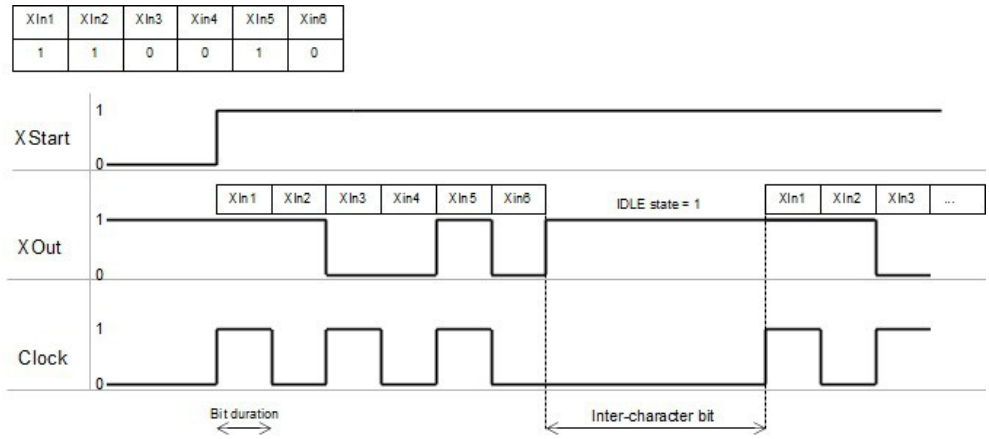
The following diagrams show the example of a 6-bit sequence to be transmitted with the two different IDLE status network settings.

<table>
<tr><th colspan="2">SYNCHRONOUS TRANSMISSION</th></tr>
<tr><td>IDLE status = 0

Number of bits to send = 6

Number of inter-character bits = 4</td><td>
</td></tr>
<tr><td>IDLE status = 1

Number of bits to send = 6

Number of inter-character bits = 4</td><td>
</td></tr>
</table>

 2024 © Pizzato Elettrica S.r.l.

Asynchronous transmission

In the asynchronous transmission, inputs have the same functions indicated in the synchronous transmission.

This type of transmission is based on the Manchester Code[1], which defines the values of bits 0 and 1 not as static levels but as status transitions: when 0 must be transmitted, an upward transition of the signal occurs, whereas when 1 must be transmitted, the coded signal undergoes a downward transition.

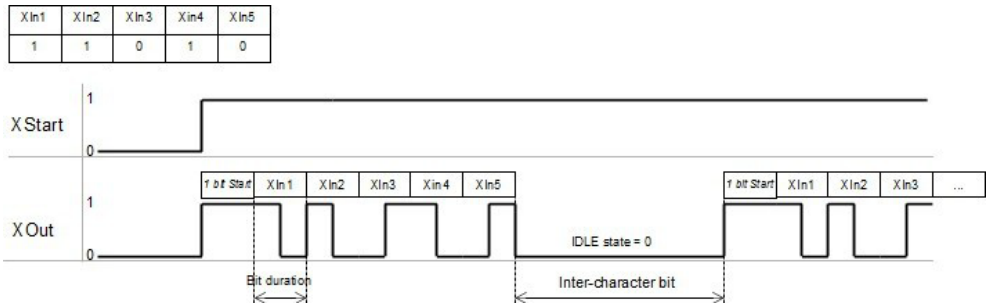
Since it is a self-synchronizing coding, the use of the clock signal (XClock) is not necessary in the Manchester Code, therefore only one output cable (XOut) has to be connected to the safety module.

At the beginning of the data transmission, a start bit is inserted, in order to signal the beginning of the sequence of data to be encoded to the receiver. Unlike the sensor status bits (XIn1, XIn2, etc.), the start bit does not execute the state transition for its entire duration, so the value of the start bit is equal to 1 with IDLE status set to 0 and equal to 0 with IDLE status set to 1.

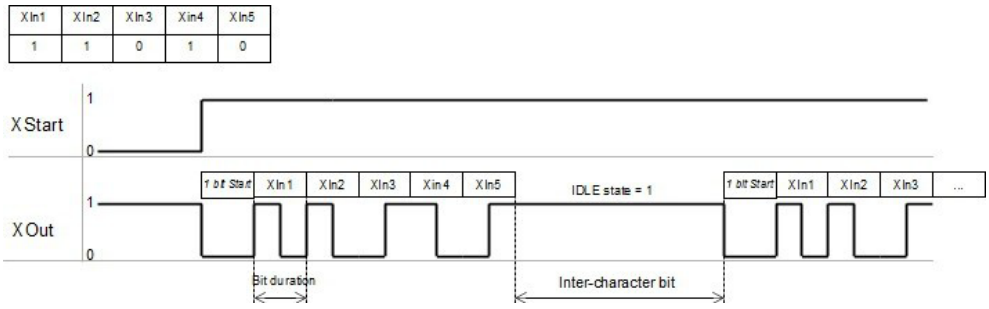
The following diagrams show the example of a 5-bit sequence to be transmitted with the different settings of the IDLE status network settings.

<table>
<tr><th colspan="2">ASYNCHRONOUS TRANSMISSION</th></tr>
<tr><td>IDLE status = 0

Number of bits to send = 5

Number of inter-character bits = 4</td><td>
</td></tr>
<tr><td>IDLE status = 1

Number of bits to send = 5

Number of inter-character bits = 4</td><td>
</td></tr>
</table>

[1] For more information see https://en.wikipedia.org/wiki/Manchester_code

 2024 © Pizzato Elettrica S.r.l.

Notes for using OS type outputs

The SERIAL functional block can be used on all the safety modules of the Gemnis series, using both signal outputs (O type) and safety outputs (OS type).

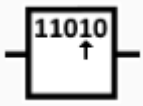
> ⚠ Even if OS type safety outputs are used, the data transmission performed by the Serial functional block must be used for signaling purposes only. The safety functions must be performed with electrical circuits and hardware and software structures that reach PL or SIL safety levels required by the technical standards EN IEC 62061 and EN ISO 13849-1 for each application.

However, if OS type safety outputs are used, the PLC may misread the module's output signal if the duration of the safe outputs' deactivation pulses (necessary for detecting faults deriving from short-circuits) is detected by the PLC program execution cycle.

If OS type safety outputs are used for transmitting the output signals of the SERIAL functional block, it is recommended to disable the test pulses through the "Proprieties" of the safety output (see 5.7.4.2), or alternatively to insert a timer filter with a set time greater than 1 ms or equal to 1 ms in the PLC program.

On the www.gemnis.com website, some examples of PLC programs are available for decoding theoutput signal from a safety module using the SERIAL functional block.

5.7.3.29 Functional block BTST

Name	BTST	Bit Test. At the output bit XOut it returns the bit value of the input at the position set in the parameters. 
Inputs	Bin or WIn	Input value for the bit test
Parameters	Input type: B W Bit position for the test:	 Type B input value (Byte) Type W input value (Word) From 0 to 7 (Byte) From 0 to 15 (Word)
Outputs	XOut	Tested bit value

The BTST functional block allows to check the value of a specific bit in a input number. The input value can be a type B or W.

The position of the bit to be checked can be set through the parameter in the "Properties" window.

Example:

- Type B input value = 137
- Bit position = 3

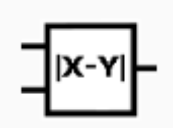
$$(137)_{10} = (1000\ 1001)_2$$

1	0	0	0	1	0	0	1
7	6	5	4	3	2	1	0

- XOut value = 1

 2024 © Pizzato Elettrica S.r.l.

5.7.3.30 Functional block ADIFF

Name	ADIFF	Absolute difference mathematical function 
Input	BIn or WIn (BRef or WRef)	Input with the minuend value (Optional) Second input with the subtrahend value
Parameters	Data type: B W Subtrahend: Constant Input Show operation sign	 Type B input (Byte) Type W input (Word) The subtrahend is a constant value The subtrahend is the second input BRef or WRef Enable the output bit XOver
Outputs	BOut or WOut (XOver)	Absolute difference result (Optional) sign of the difference

The functional block ADIFF makes the absolute difference between two values. Minuend and subtrahend can be B or W type, changing the settings in the Properties window.

In the Properties window it is possible to set if the subtrahend is a constant value or a second input BRef or WRef, which will be enable.

The operation result in absolute value is in the output BOut or WOut; it is possible to have information about the sign of the result enabling the output bit XOver, which will be 1 in case the second input (BRef or WRef) is greater than the first input (negative sign).

Example:

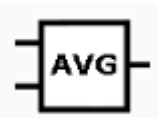
- Type B input BIn = 25
- Constant subtrahend = 32

25 – 32 = -7

- BOut = 7
- XOver = 1

 2024 © Pizzato Elettrica S.r.l.

5.7.3.31 Functional block AVG

Name	AVG	Arithmetic mean between two values 
Inputs	BIn or WIn (BRef or WRef)	Input value (Optional) Second input
Parameters	Data type: B W Mean: Constant Input	 Type B input (Byte) Type W input (Word) The mean is with a constant value The mean is with the second input BRef or WRef
Output	BOut or WOut	Arithmetic mean result

The AVG functional block make the arithmetical mean between two values. In the properties window it is possible to set if the mean is with a constant value or with a second input. Operands can be B or W type.

The output BOut or WOut show the mean result, truncated to the integer value.

Example:

- Type B input BIn = 11
- Mean with constant = 4

$$\text{Mean value} \rightarrow \frac{11+4}{2} = 7{,}5$$

- BOut = 7

 2024 © Pizzato Elettrica S.r.l.

5.7.3.32 Functional block WTOB

Name	WTOB	Conversion from W (word) to B (byte) 
Input	WIn	W type input
Parameters	Bits shift	From 0 to 8
Output	BOut	B type result from the conversion

The functional block WTOB allows to convert an input W type value to a B type value. The conversion it is made by truncating first 8 significant bits of the input WIn. In the properties window it is possible to set the number of bits N for the logical right shift before truncating.

The logical right shift moves the bits to the right by removing the first N less significant bits and by adding N bits equal to 0 on the left.

Example:

- Input value WIn=3045
- Number of bit for the logical right shift = 3

$$(3045)_{10} = (0000\ 1011\ 1110\ 0101)_2$$

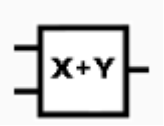
$$0000\ 1011\ 1110\ \underbrace{0101}_{\text{3 bits shift}} \rightarrow \underbrace{0000\ 0001}_{\text{Truncated}}\ \underbrace{0111\ 1100}_{\text{Result Byte}}$$

$(0111\ 1100)_2 = (124)_{10}$

- BOut = 124

 2024 © Pizzato Elettrica S.r.l.

5.7.3.33 Functional block SUM

Name	SUM	Addition of two values 
Inputs	BIn or WIn (BRef or WRef)	First addend value (Optional) Second addend value
Parameters	Data type: B W Addend: Constant Input Show the overflow	 Type B input (Byte) Type W input (Word) The second addend is a constant value The second addend is an input value BRef o WRef Enable the output bit XOver
Output	BOut or WOut (XOver)	Addition result (Optional) addition overflow

The SUM functional block allows to make the math addition of two values. The addends can be type B o type W, changing the settings in the Properties window.

In the Properties window it is possible to set if the second addend is a constant value or a second input BRef or WRef, which will be enable.

The operation result is in the output BOut or WOut.

The maximum value that can be represented with N bits (2^N-1), therefore for a type B (8 bits) is 255, while for a type W (16 bit) is 65636. The binary representation of these two values is:

B → $(255)_{10} = (1111\ 1111)_2$

W → $(65535)_{10} = (1111\ 1111\ 1111\ 1111)_2$

If we add 1 to both vales, we will have:

B → $(1111\ 1111)_2 + (0000\ 0001)_2 = (1\ 0000\ 0000)_2$

W → $(1111\ 1111\ 1111\ 1111)_2 + (0000\ 0000\ 0000\ 0001)_2 = (1\ 0000\ 0000\ 0000\ 0000)_2$

In this case both results are impossible to be represented with the available bits. In fact, the results are 256 (impossible to represent in 8 bits) and 65536 (impossible to represent in 16 bits).

Therefore, it can happen that the operation generates a result which cannot be represented in the output number of bits. In this case we have an overflow. It is possible to identify the overflow with an optional output bit called XOver, which will be 1 in case of overflow and 0 in the opposite case.

The output vale BOut or WOut in case of overflow will be truncated removing the most significant bit, therefore the decimal result will be the addition result minus 2^N (with N number of bits).

Example:

- Type B input BIn = 12
- Constant addend = 5

12 + 5 = 17

- BOut = 17

 2024 © Pizzato Elettrica S.r.l.

- XOver = 0

Example with overflow:

- Type B input BIn = 175
- Constant addend = 167

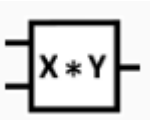
$$175 + 167 = 342$$

The maximum representable value for a B type (8 bit) data is 255 → overflow

In this case the BOut result will be represented as the addition result minus 2^N, with N=8:

- BOut = 342 - 2^8 = 342 – 256 = 86
- XOver = 1

5.7.3.34 Functional block MUL

Name	MUL	Multiplication math function 
Inputs	BIn or WIn (BRef or WRef)	Input value (Optional) multiplier second input
Parameters	Data types: B W K α Multiplier types: B * α = B B * B = B B * K = W B * B = W W * α = W W * B = W Shows the overflow	 Type B input (Byte) Type W input (Word) Integer multiplier Decimal multiplier Multiplication of a type B input and a constant α with a type B result Multiplication of two type B inputs with a type B result Multiplication of a type B input and a integer constant K with a type W result Multiplication of two type B inputs with a type W result Multiplication of a type W input and a constant α with a type W result Multiplication of a type W input and a type B input with a type W result Enable the output bit XOver
Output	BOut or WOut (XOver)	Multiplication result (Optional) overflow on the result

The MUL functional block makes the multiplication math function of two values. In the properties window it is possible to choose different type of multiplication:

- B * α = B

 2024 © Pizzato Elettrica S.r.l.

- B * B = B
- B * K = W
- B * B = W
- W * α = W
- W * B = W

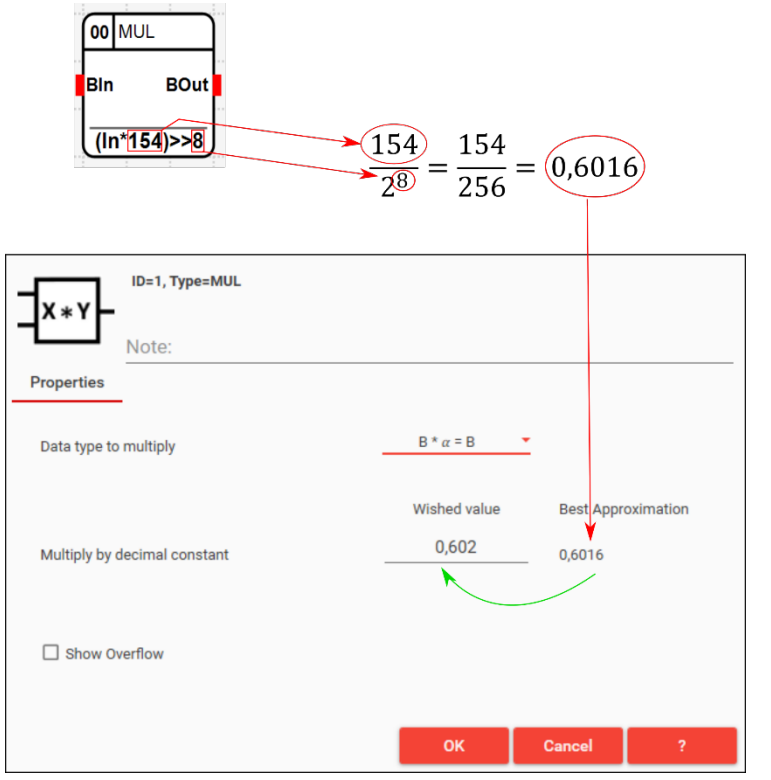
B * α = B

The functional block makes the multiplication of a type B input and a constant value α. The value of the constant α can be a number from 0,000 and 255,000, and it is possible to set the value in the Properties window.

Please note that the multiplication by a decimal number is approximated through the multiplication by an integer number and dividing by power of 2 (bit right shift).

For example, the multiplication by a constant α = 1,602 (desired value), the best approximation is the multiplication by 158 and dividing by 2^8; in this case BIn will be multiplied by 0,6016 which is the best approximation of the desired value.

It is not necessary that the user makes this calculation. In the Properties window there is a filed for the desired value, Gemnis Studio will calculate automatically the best approximated value.



The type B result, truncated of the decimal part is in the output BOut.

In case it is not possible to represent the result in a type B number (greater than 255), it will be truncated keeping the first 8 less significant bits. For example:

- BIn = 205
- α = 1,5

205 x 1,5 = 307,5 (where the decimal part will be truncated, therefore it remains 307)

 2024 © Pizzato Elettrica S.r.l.

$$(307)_{10} = (1\ 0011\ 0011)_2$$

BOut is a type B number; therefore, the result will be truncated keeping the first 8 less significant bits:

$$(1\ 0011\ 0011)_2 \rightarrow \text{truncated } (0011\ 0011)_2$$

$$(0011\ 0011)_2 = (51)_{10}$$

BOut will be equal to 51 and XOver (if enabled) will be equal to 1.

B * B = B

The functional block makes the multiplication of two type B inputs: BIn and BRef. The result is a type B value and is reported at the output byte BOut.

In case it is not possible to represent the result in a type B number (greater than 255), it will be truncated keeping the first 8 less significant bits. For example:

- BIn = 205
- BRef = 3
- Bit to shift = 0

$$205 \times 3 = 615$$

$$(615)_{10} = (10\ 0110\ 0111)_2$$

BOut is a type B number; therefore, the result will be truncated keeping the first 8 less significant bits:

$$(10\ 0110\ 0111)_2 \rightarrow \text{truncated } (0110\ 0111)_2$$

$$(0110\ 0111)_2 = (103)_{10}$$

BOut will be equal to 103 and XOver (if enabled) will be equal to 1.

Inside the properties window it is possible to set the number of bits for the logical right shift of the multiplication result, before the truncation.

The N bits logical right shift moves the bits to the right, deleting the first N less significant bits and adding N bits equal to 0 on the left.

The previous example, setting a bit shift of 2 becomes:

- BIn = 205
- BRef = 3
- Bit to shift = 2

$$205 \times 3 = 615$$

$$(615)_{10} = (10\ 0110\ 0111)_2$$

$$\text{Bit shift di 2 bits } (10\ 0110\ 01\cancel{11})_2 \rightarrow (1001\ 1001)_2 = (153)_{10}$$

 2024 © Pizzato Elettrica S.r.l.

BOut is a type B number; therefore, the result can be represented without truncation.

BOut will be equal to 153 and XOver (if enabled) will be equal to 0.

B * K = W

The functional block makes the multiplication of a type B input BIn by an integer constant K. The constant K can be an integer number from 0 to 255. The result is a type W value and is reported at the output byte WOut.

B*B = W

The functional block makes the multiplication of two type B inputs: BIn and BRef. The result is a type W value and is reported at the output byte WOut.

W * α = W

The functional block makes the multiplication of a type W input by a constant value α. The value of the constant α can be a number from 0,000 and 255,000, and it is possible to set the value in the Properties window. More details can be found in the previous section **B * α = B**, the only differences are the type of input and output that here are type W (WIn and WOut).

W * B = W

The functional block makes the multiplication of two inputs, type W and type B: WIn and BRef. The result is a type W value and is reported at the output byte WOut. More details can be found in the previous section **B * B = B**, the only differences are the type of input and output that here are type W (WIn and WOut).

 2024 © Pizzato Elettrica S.r.l.

5.7.4 Outputs

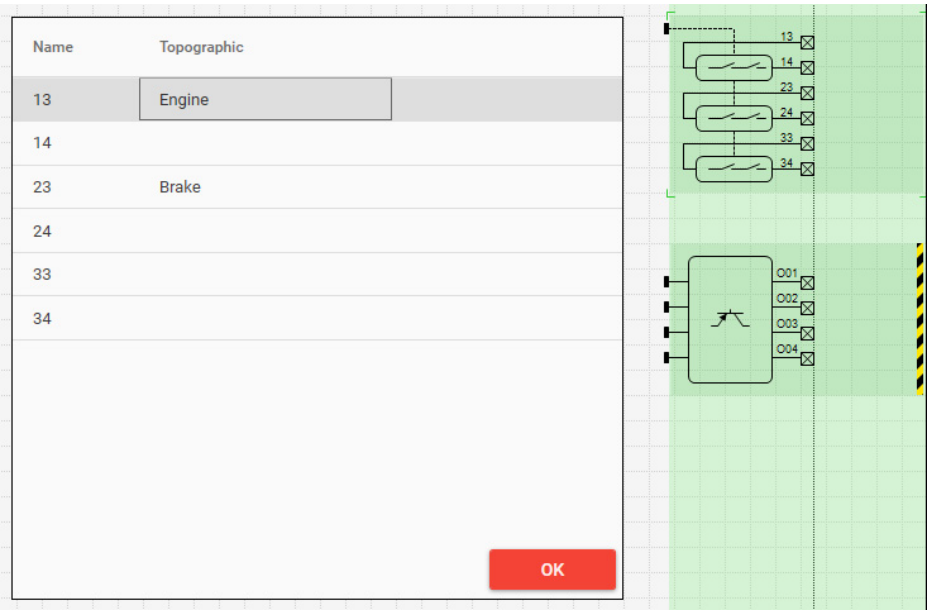
When a project has been created, depending on the module type, in the outputs area are automatically placed objects that represent safety outputs and signal outputs of the module. If the outputs are not connected via connectors, they remain off.

5.7.4.1 Safety Outputs

The safety outputs are represented within green rectangles. Two different types of safety output are available: electronic or relay.

The relay safety outputs are represented by the presence of a connection point that goes to act on all available contacts of the relay. This highlighted the mechanical connection between the various contacts. Relays physically present within the module are always in pairs, piloted separately by the two-processor module, and connected in series as evidenced from the drawing which represents contact pairs in series. In case there are multiple relay outputs (relay pairs) more connection points will be available, each one that refer to different groups of contacts. The electrical connections of relays individual contacts are represented through the clamps, with their codes, on the dashed line right border that represents the module. The safe output area extends further to the right to give the ability to add notes on the clamps.

Clicking with the right mouse button over the safe outputs area you can select the Properties tab (see figure) where you can enter the desired text.



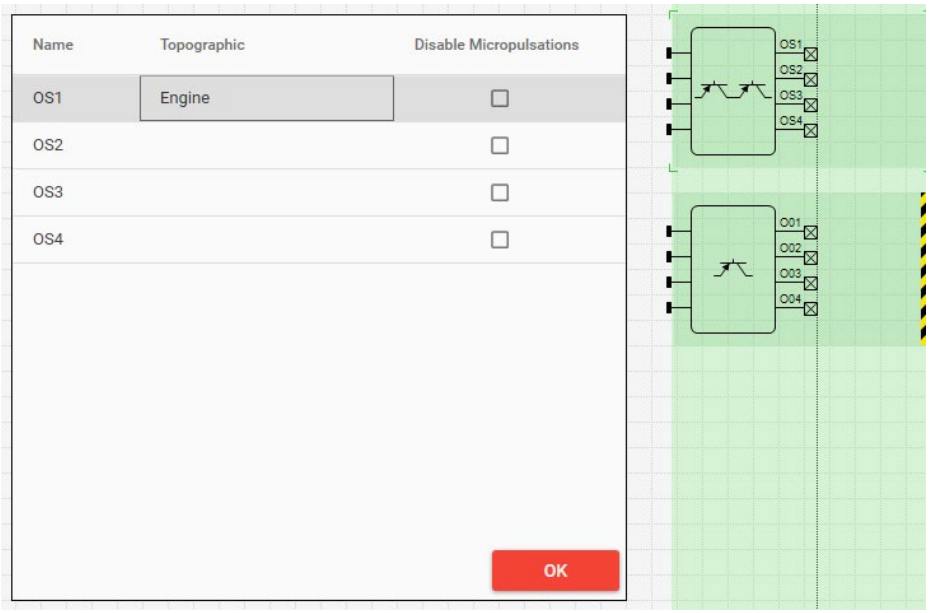
These descriptions are automatically included in the list of electrical connections (see 5.7.1). As you can imagine to enable a relay safety output and all relevant contacts, just connect the safety output connection point with the output of a functional block or a sensor.

Please note that the power supply for relay is taken from "24V" and "0V"terminals and that contacts have always to be protected using external fuses.

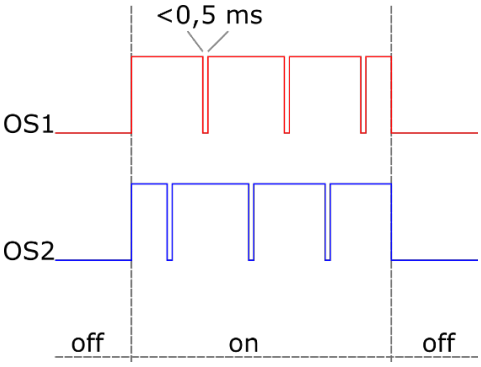
Electronic safety outputs are likewise represented within a frame of the same color and by the presence of one or more connection points, each corresponding to a pair of transistors. Each output of this type is physically represented by two transistors separately controlled by the two-processor module, and connected in series between the clamp "24V" and clamp "OSx". The electrical control is done with reference to the clamp "0V". Unlike the contacts of safety relays, transistor outputs are always independent of each other. Even in this case you can add comments to the safety output clamps and such descriptions are automatically included in the list of form electrical connections (see 5.7.1). Terminals

 2024 © Pizzato Elettrica S.r.l.

"24V" and "0V" are present only for clarity of electrical connections and cannot be commented.



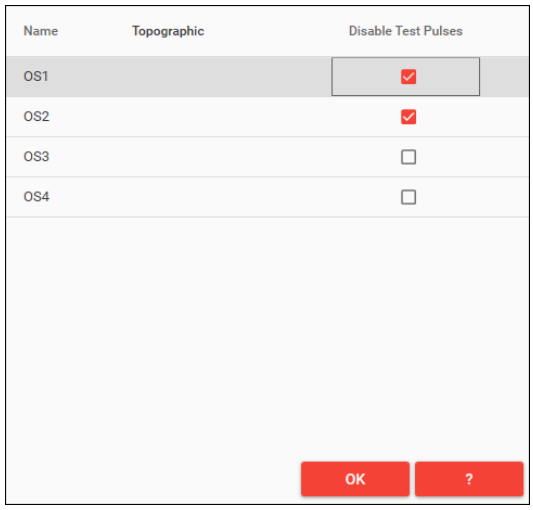
5.7.4.2 Test pulses



Please note that electronic safety outputs are internally protected against short-circuit and that mechanisms are implemented to control continuously the integrity of the transistor output to avoid short circuits to power supply or to other safety outputs. To carry out these controls in every safety output are created test pulses (small breaks which stay for typically less than 0.5 ms, see technical specifications) when the outputs are active. These micro-interruptions are not influent on passive loads (contactor or similar) but can be read, and false the reading, in case the outputs are connected to a PLC or electronic motion controllers. If these outputs are connected to the inputs of another Gemnis module there are no problems because all inputs of the modules of this series are filtered against micro- interruptions.

A new project has always enabled this test pulses by default, but if necessary it is possible to disable through the properties window (right click on the outputs area).

 2024 © Pizzato Elettrica S.r.l.



Disabling the test pulses decreases the diagnostic coverage on the safety outputs, especially the short-circuits to the positive signal can be detected only at the deactivation of the safety outputs:

	Failure modes on the electronic safety outputs OSx		
Test pulses	Short-circuit to the positive signal of the power supply 24V DC	Short-circuit to ground	Short-circuit between two outputs
Enabled	Detectable	Detectable when the output is ON	Detectable (even in case one of them has the pulses disabled)
Disabled	Detectable when the output is OFF	Detectable when the output is ON	Detectable only when the two outputs have different values

In case of disabling of the test pulses, the timings for the fault detection depends on the logic with which the safety outputs are activated. In case the state of the safety outputs remains the same very long the diagnostic coverage on the safety outputs is limited at:

$$DC \geq 60\ \%$$

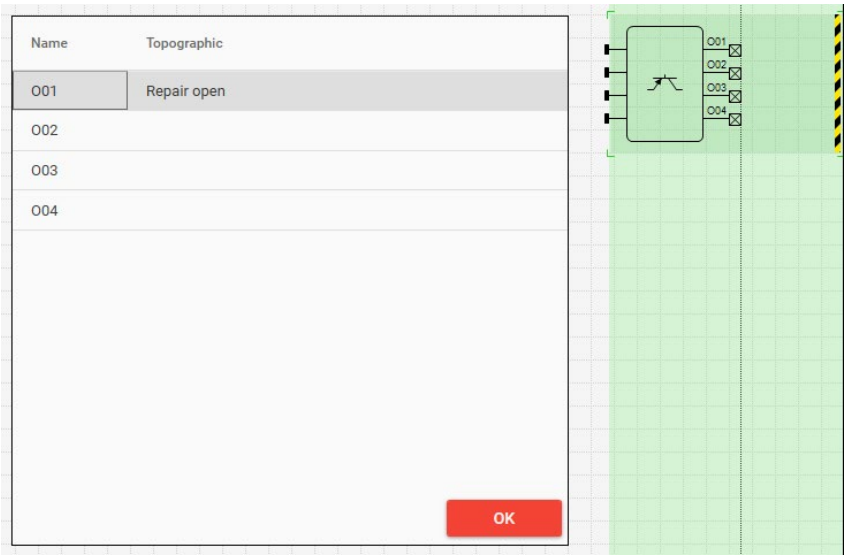
It follows that in these cases the maximum reachable level of safety is PL d / SIL 2.

However, a specific analysis (fault exclusion, frequent activations, etc.) could bring to better results.

5.7.4.3 Signal outputs (unsafe)

Non-safety outputs are marked by a green color area with a yellow/black dashed line on the right side. This dashed line is used only to remind the unsafe nature of such outputs (exactly as previously seen for some Sensors). For each transistor is represented a connection point that may be associated with the output of a functional block or directly with the output of a sensor. Even in this case you can add a comment to each output terminal and such descriptions are automatically included in the list of electrical connections (see 5.7.1). To do this simply click on the green area with the right mouse button and select the Properties tab (see image below).

 2024 © Pizzato Elettrica S.r.l.



Unlike safe outputs, signal outputs take their power from clamp “A1”, and refer to clamp “A2” (ground) and share their current with the same line that feeds the Test signals (Txx terminals). This line is controlled from a current limiter which, in case of overload (see technical specifications), blocks the current and put the module in ERROR state.

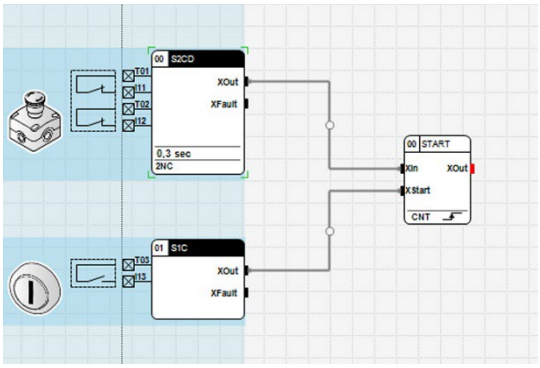
The signal outputs are not subject to micro-interruptions as in the case of electronic safety outputs.

5.7.5 Connectors

Connectors are objects that connect the Sensor outputs with inputs of Functional blocks and their outputs with the inputs of other Functional blocks and so on until the connection points of Outputs and in this way create the logical structure of the Application Program.

The connector "transmits" or "carries" the information from a connection point output of an object in a connection point entry of another object.

The connectors always start from an output connection point (of a Sensor or a Functional Block) and ends in an input connection point (of a Functional Block or Output). To create a connection, simply click on the starting point of connection and, holding down the button, move the mouse over the target connection point, and then release the mouse.



The connection thus created is presented in its standard version as a "line" (there are other versions of graphics connector, see below) that connects the two connection points. This line can be direct or broken in which case has at its middle a small pane that lets you move the line (if you wish) optimizing its positioning point. Each connection can be cancelled by clicking on it and pressing the delete key on your keyboard or clicking on it with the right mouse button and then selecting Delete.

 2024 © Pizzato Elettrica S.r.l.

Some operations with the connectors can't be performed as it would lead to a situation of non-interpretability or instability of the program and Gemnis Studio would blocks they immediately. These operations are:

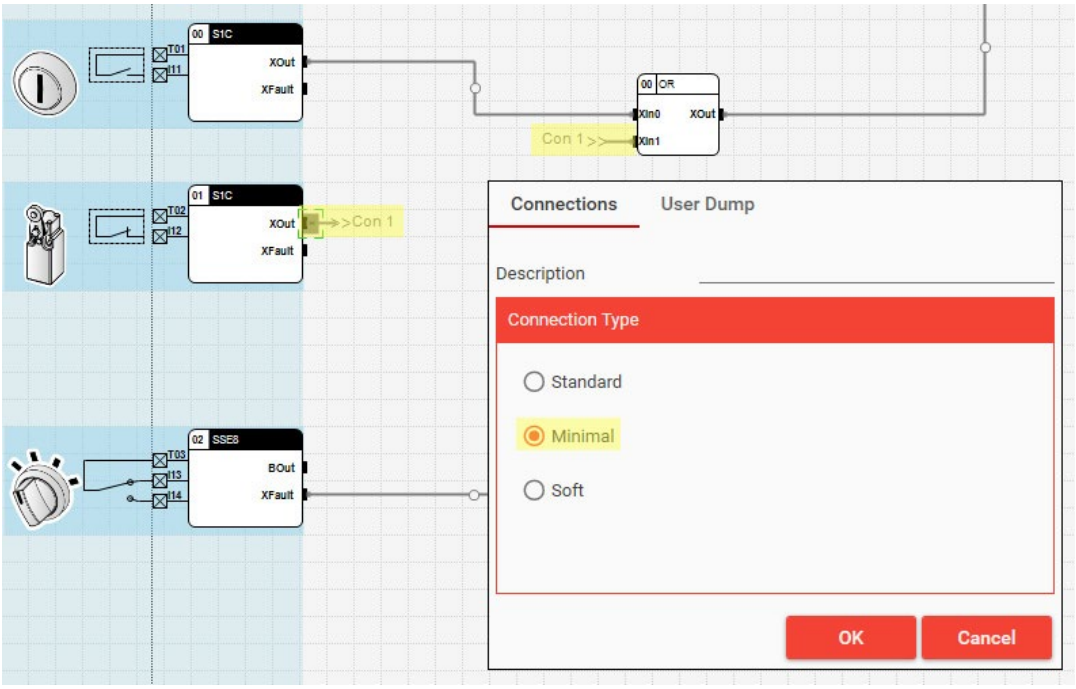
- Connect connection points having underlying data of a different type (X, B, W). For example, you cannot connect an output of type X with an input of type B as Gemnis does not allow implicit conversion of different data types.
- Create connectors with one end not connected.
- Create connectors that connect inputs with inputs or outputs with outputs.
- Connect multiple connectors from various outputs in the same input (electric OR). If such an operation was needed it is necessary to insert an intermediate functional block, e.g. a logical OR.
- Create connectors that connect the output of a functional block with an input of the same functional block. These connections may lead to possible "oscillation" situations and are prohibited.
- Create a sequence of connections which create a ring between two or more functional blocks. For example, given two functional blocks FB1 and FB2, it is forbidden to connect the output of FB1 with an input of FB2 and then output of FB2 with an input of FB1.

Standard connectors consist of straight lines. There is the possibility of having other graphic formats for the connectors. To change the graphical format of a connector just select it by clicking on the intermediate rectangle or on connection points where the connector ends and select Properties.

 2024 © Pizzato Elettrica S.r.l.

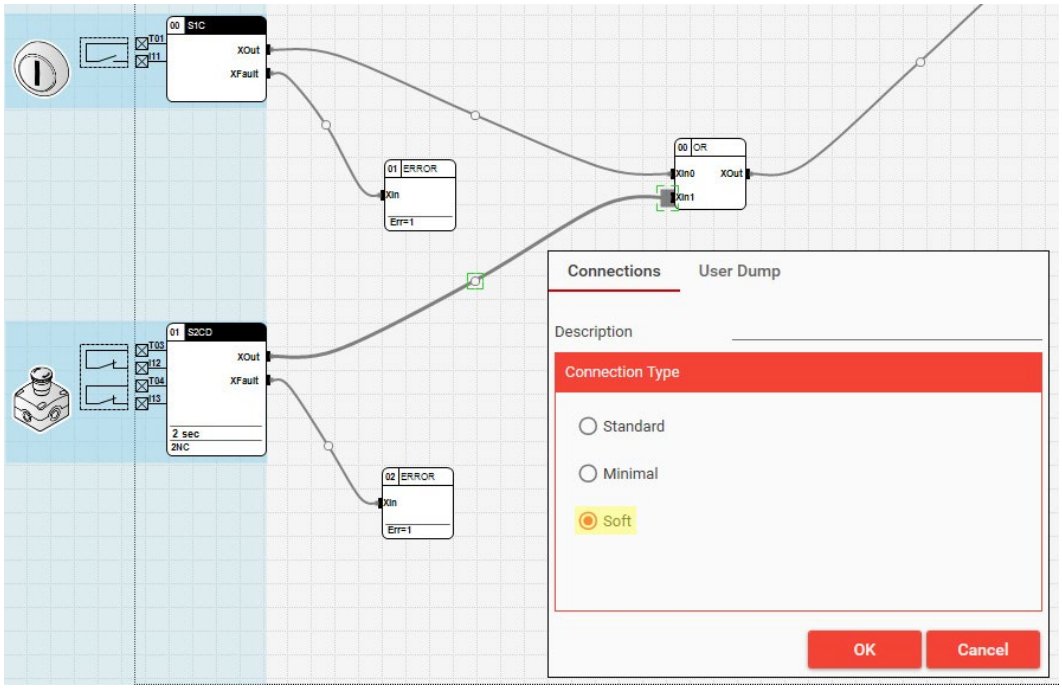
"Minimal" graphic format

In this graphic format the connector is not represented, but are only created folding arrows and a reference text. These connections are useful for functions that do not want to give evidence or to group functional blocks in other locations.



"Soft" graphic format

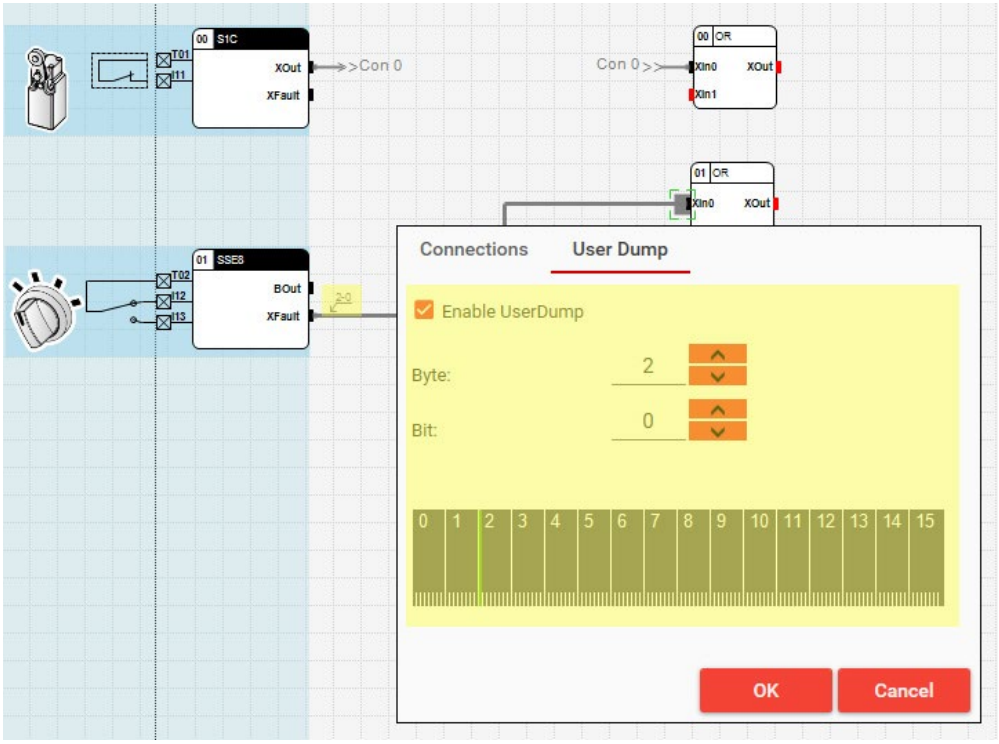
In this graphic format the connector is represented by a curved line.



 2024 © Pizzato Elettrica S.r.l.

5.7.5.1 Connector controlled by User Dump

The State of a connector at run time can be controlled externally via a User Dumps command (see 0). To indicate whether a connector must be checked, and in what position of the output string the connector status must be inserted, it is used the “User Dump" tab in the properties of the connector.



As you can see in the picture to activate the control you need to set the flag "Enable UserDump" and then indicate which position of the string to insert the connector status. If the data type of connector is X you have to indicate the position of the bit in the string, if the data is of type B will have to indicate the byte position and if the data is of type W, the position of word (2 bytes). The "ruler" in the form highlights with a green colored bar the insertion point of data in the string. If some positions are already occupied they are yellow colored. In the case of overlapping with other data the bar becomes red. Note that bits in the ruler are represented for convenience in reverse order compared to the real position of the bit in the string, i.e. bit 0 in the ruler is located at the left end of a byte, while in reality it is in the right end.

When a connector is controlled by User Dump a graphic symbol appears next to the starting point of the connector. It is a small arrow and two data indicating respectively the Bytes and bits of the given position within the string of User Dump. For example, the string "1-0" indicates that the value of the connector is inserted into bit 0 of byte 1 of the string.

If from a connection point stars more connectors you just activate the User Dump only on one of them because they all report the same information.

You cannot refer to User Dump an output connector from a label. But, because the label merely replicates the given input connector, if you plan to check it just control the input connector.

If the User Dump Tab does not appear among the properties of the connector be careful not to have selected more than one connector at a time.

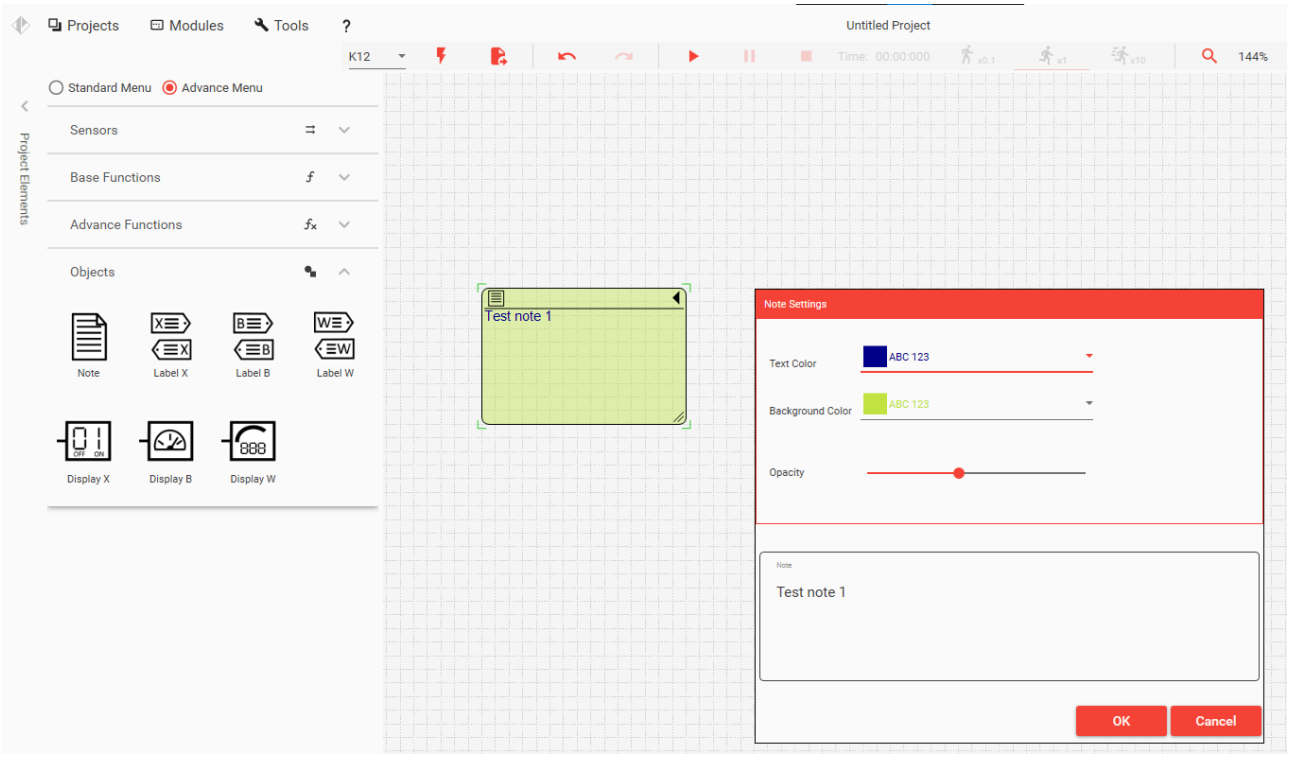
 2024 © Pizzato Elettrica S.r.l.

5.7.6 Other items

Objects tab inserts several software features that are not related to those of the Sensors or Functional Blocks. In this release of Gemnis Studio are objects "Note" and "Label".

5.7.6.1 Note

It is possible to insert directly into the desktop notes through the Note object present in the Desktop Objects Tab.



Notes can be positioned as you want on desktop and, in addition to text, you can change the text color, background color and transparency of every single note. The frame size of the note can be changed by dragging the symbol at the bottom right corner of the note. Finally, the note can be collapsed/expanded by clicking on the arrow in the upper-right corner of the box.

Default values for color and transparency of the note can be set from the Options menu (see General options).

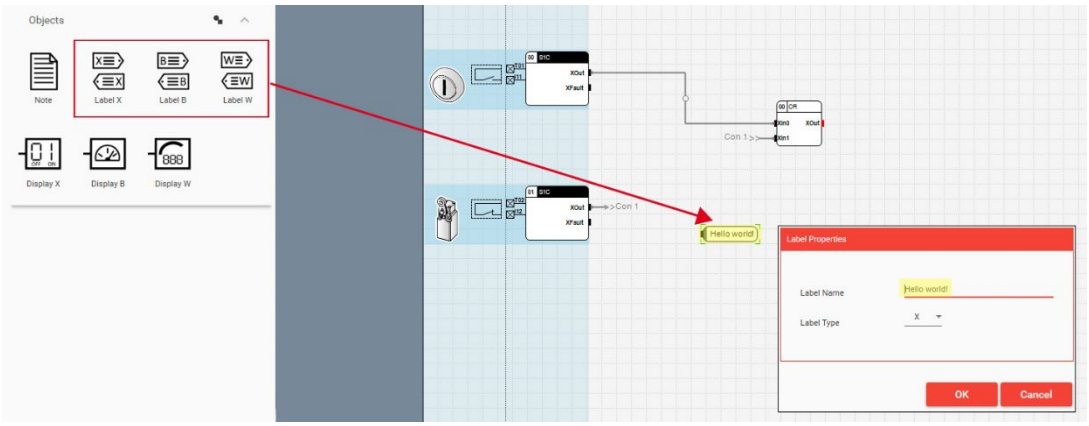
 2024 © Pizzato Elettrica S.r.l.

5.7.6.2 Labels

The object label is a very powerful tool to create multiple connections, simplify the display, and help to identify intermediate functions of the project, especially in the case of projects that span over multiple pages.

The label can be thought of as a Functional Block of "identity" type, where each output corresponds to the single input of the block, with the peculiar characteristic that the functional block can be graphically split to be designed in the proximity of the point of interest.

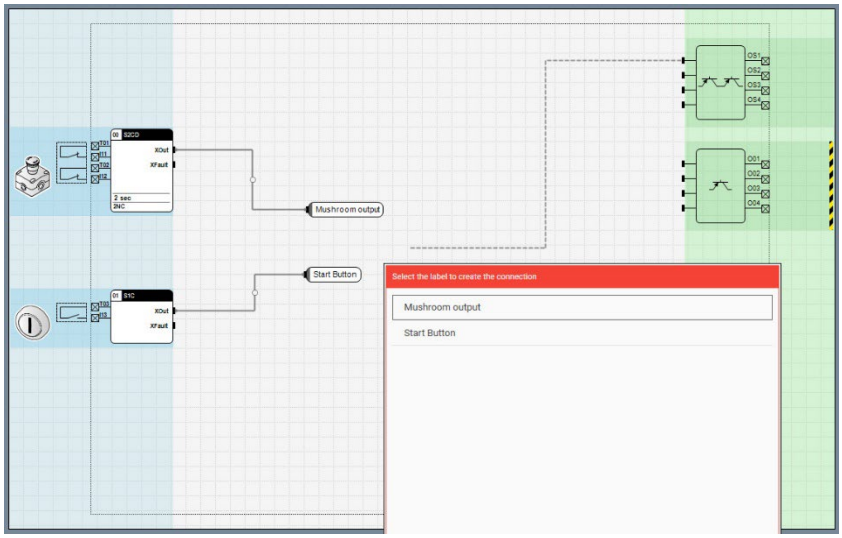
To create a label, simply drag the "Label" object from Desktop Objects tab.



Clicking with the right mouse button on the label, you can change the text or label name.

The label thus created shows a connection point of the input that can be connected to the output of any sensor or functional block of the same type (X, B or W).

At this point, you can create a connection to the same label, creating a connection that starts from an input connection point, and then releasing the mouse in a free area (easier to make than to say). If you have created a single label it will automatically connect. If there are multiple labels it will be prompted with which of them you intend to make the connection (see figure below).

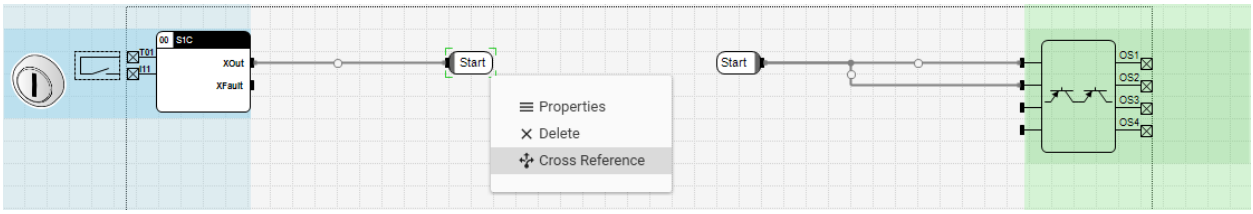


 2024 © Pizzato Elettrica S.r.l.

Labels can be renamed at any time. Connection points of labels are normal connection points, and then from a label can also leave more outbound connections as in the following example.



If labels are spread all over the desktop, the Cross Reference function can help you to find them out. This function can be shown with a right click on the label.



Clicking on it the program will execute a desktop search for all the labels with the same name and the outcome will be displayed on the left bottom of the screen into the log area.

Clicking on a result will automatically select the label, and the desktop will be positioned to be centered on the object. The identification of this is highlighted with a colored rectangle that gradually disappears

 2024 © Pizzato Elettrica S.r.l.

5.7.6.3 Display

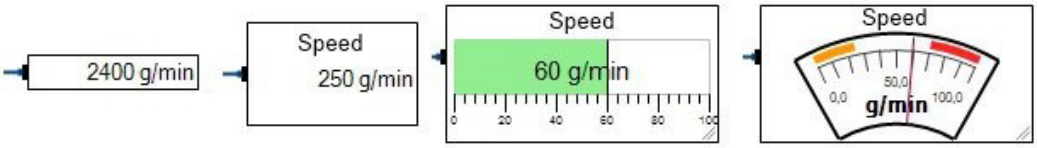
The Display object is useful to see more clearly the value of some variables during the simulation or monitoring. Allows you to display the value of variables of type X, B, or W in different formats. It 'also very useful if you do not want to use some output of a functional block. Since the compiler does not accept that the functional blocks have unconnected connection points, if any output is not necessary, or need to be controlled by userdump function (and therefore need a connector) you can connect this output to a Display object.

These objects, depending on the type of the input, have different aspects selectable from the object menu.

- In case of digital input, the option of display visualization are four: the first (minimal) shows the value, optimizing the space in the desktop, the second (numeric) shows with a digital number the input value and the eventual label assigned by the user, the third (Green Led) shows a diode led that turns green or grey depending on the state on or off, then the last, the forth (Red Led) is represented by the same symbols except that the LED is red instead of green.



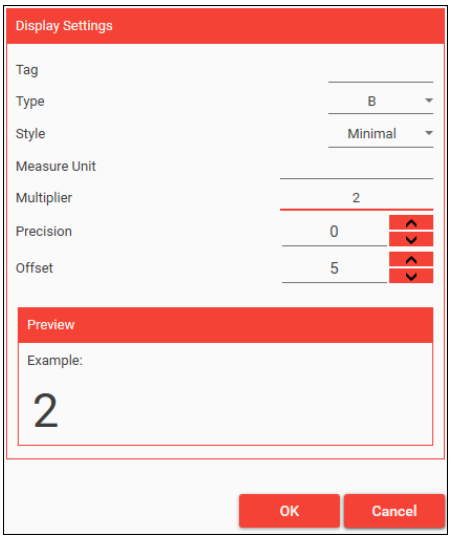
- If the input is a Byte (B) or Word (W) type, the control can take the following graphical aspects: the first (minimal) simply shows in the desktop the value and the units (multiplied for the coefficient set), the second (numeric) is similar to the first but showing also the label, the third and the fourth (Analogic and Digital) much more attractive represent respectively a classic analogic instrument with pointer and a digital indicator bar. In this last two controls you have to set the minimum and the maximum level and you can set also the minimum and maximum attention level. This is useful if the user wants to have an immediate visual feedback of the assumed values of the variables.



If Byte (B) or Word (W) type display are used, in the Properties menu it is possible to set numerical values called Multiplier and Offset in order to process the reading value of the sensor and to return an indicative value of the physical quantity to be measured.

The sensor reading value is exclusively processed in a linear manner.

 2024 © Pizzato Elettrica S.r.l.



Setting the Multiplier and Offset is very useful, for example, when using current transducer sensors, for which the minimum value of output current is greater than zero Ampere. It is then possible to detect any faults deriving from an interruption of the electrical circuit.

For example, if it is necessary to measure the angular rotation of a moving part connected to an electric motor, with a sensor in a current with a range of output values between 4 mA and 20 mA.

The correspondence between the sensor output value and the position of the moving part is summarized in the following table:

Output value from the current transducer	Rotation of the moving part
4 mA	0°
20 mA	90°

To ensure that the Display object of Gemnis Studio, associated with the 4 - 20 mA current sensor, returns the correct rotation value of the moving part, it is necessary to set the Multiplier and Offset values in the Properties menu. They must be calculated with the linear algebra formula that identifies the equation of a straight line passing through two points in the Cartesian plane:

$$\frac{x - x_1}{x_2 - x_1} - \frac{y - y_1}{y_2 - y_1}$$

If the x-axis indicates the current value of the transducer and the y-axis indicates the rotation of the moving part, the previous equation becomes:

$$\frac{x - 4}{20 - 4} - \frac{y - 0}{90 - 0}$$

from which the straight-line equation is obtained explicitly

$$y = 5{,}625\, x - 22{,}5$$

The Multiplier value corresponds to the angular coefficient of the straight line, that is + 5.625, while the Offset value corresponds to the ordinate at the origin, ie - 22.5.

 2024 © Pizzato Elettrica S.r.l.

5.7.7 Simulation

Gemnis Studio is equipped with a very useful simulation environment to perform “virtual tests” on the program that you're making and verify proper operation before installing the program on the module.

> ⚠ Please note that the tests carried out on the Simulator does not replace real tests that should be done with the module programmed before putting machinery into service.

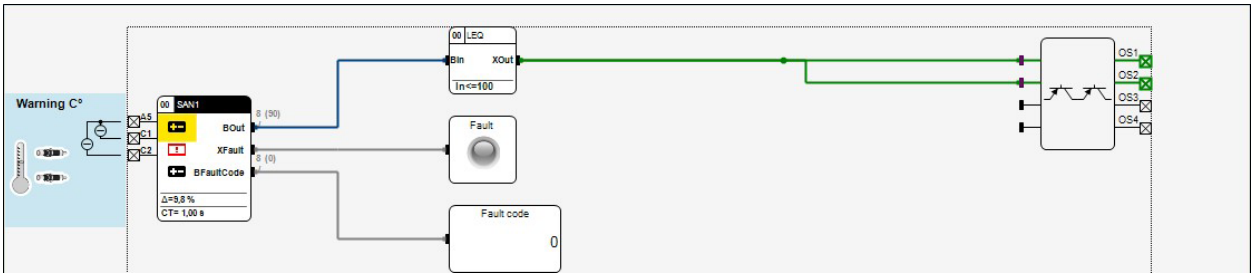
All you need to do to start the program simulation under development is to press the Start key on the button bar at the top of the desktop. It is not necessary to compile the program first, as the compile operation is invoked automatically by pressing the Start key. If the program cannot be compiled, the simulation won’t start.

Starting the simulation phase transforms the desktop and the possibilities of interaction with it. During the simulation the desktop area is locked and it is no longer possible to add, remove, modify or move Sensors, functional blocks, connections and outputs. During this phase, you can only simulate the operation of the module by interacting with the Sensors and simulating real-world conditions or operations. Specifically, the sensors expose at this stage a colored rectangular area around the "outside" part of the sensor (the part outside the dotted line representing the module) that serves as the "input" area of the main function of the sensor. For example, in the case of a "button" type sensor the area can be clicked in two ways;

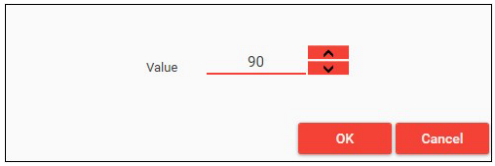
Left-clicking on it will make the pushbutton act in a stable way. It will make the contacts commute from open to close and vice versa if it is clicked again.
At the same time the sensor will switch its XOut status from 0 to 1 and vice versa if it is clicked again.
In the case of a modal switch with multiple positions, each click advances the selector position and once you reach the end it starts again.

Right-clicking on it will make the pushbutton act in a momentary way. It will make the contacts commute only as long as the pushbutton is being clicked. At the release the contacts will get back to the initial status. The same principle is applied to the XOut of the sensor (This mode of operation is available only on XOut sensors)

During simulation sensors with a BOut and/or WOut sensors works in a different way.

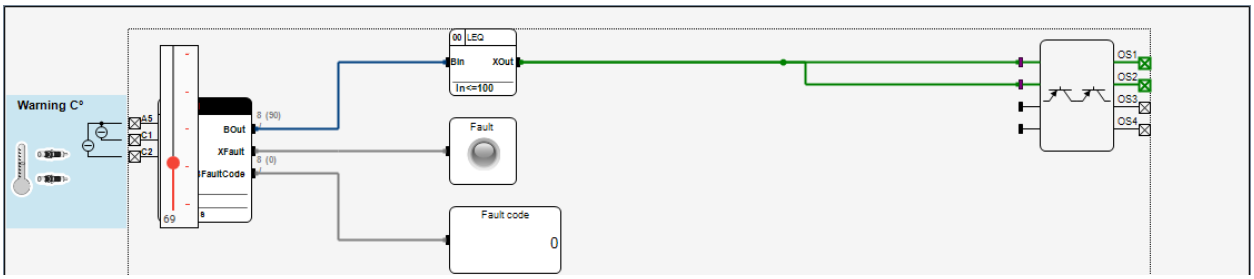


By clicking with the left pushbutton of the mouse on the highlighted icon the software will allow you to set manually the BOut value (see picture below)



On the other hand, by keeping clicked the icon with the right mouse button the software will show a slider. The cursor indicator can be dragged along the line, and a preview of the value that will be set is displayed under it (see picture below). The value will be set at the release of the mouse.

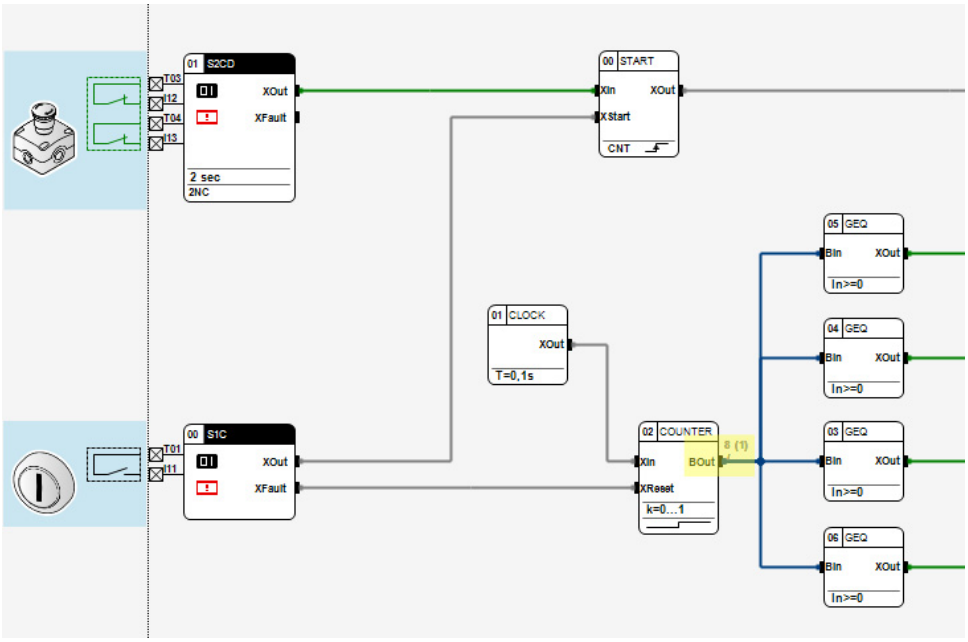
 2024 © Pizzato Elettrica S.r.l.



Each of these interactions modify sensor output variables through the connectors, which will become the input variables of the functional blocks, that evaluate them, and so on until such data will arrive to output which will activate or not, simulating exactly what will happen in the module.

The transmission of information through the connectors is made visible by a color change of connectors. In particular, the connectors remain normal coloring (grey) when the underlying data is 0 (for any type of connector) but change color when data is greater than zero. In this case the connector color varies depending on the data type of the connector, and the connector using data of type B and W show, near the starting point, the number of bits and the actual value of the underlying variable is enclosed in parentheses.

For example, in the picture you can see some X type connectors off (with value 0) grey color, X type connectors (with value 1) green color and a B type connector blue, whose underlying 8 bit variable, currently contains the value 2 (decimal).



 2024 © Pizzato Elettrica S.r.l.

5.7.7.1 Time in simulation

From the moment of simulation start-up, a system clock shows the time elapsed from the beginning of the simulation. At the same time activate the buttons Pause and Reset and buttons which changes the simulation speed, x0,1, x1 and x10.

At any time, you can press Pause and pause the simulation, for example to analyse calmly a particular state of the machine, or to make simultaneously changes in input sensors. If you want to resume the simulation you can press again the button Start.

Pressing the button Reset Gemnis Studio exits the simulation mode back to the software configuration.

At the start of the simulation, when the x0,1 button has been selected, the simulation process does a general calculation of all variables and functional blocks sensors and an update of screen values of all variables at each step of the simulation. The label x0,1 indicates that the on-screen simulation is represented with a speed of about one tenth of the actual speed, where "about" is highly dependent on computer resources where Gemnis Studio is installed and the number of active programs. For a precise statement of simulated time you only have to evaluate the data from the watch values of the simulator. Selecting the x1 box the simulation process is sped up because the screen update is performed every 10 cycles of general recalculation. That is, while the internal variables are recalculated normally in steps of 10 ms, the screen update of these values is only performed every 100 ms. In this case the simulated time becomes similar to the real-time, with the limits previously indicated. Finally, selecting the x10 the screen update is made every 100 simulation steps (1 second). Note that, in cases x1 or x10, system clock is updated at video with the same frequency and then keep in mind that, even if not seen to update the figures of hundredths or tenths of a second clock, the system is always simulating the module with steps of 10 ms.

5.7.7.2 Start conditions and special states of sensors

At the start of simulation, each sensor starts in a default state, depending on the type of sensor. For example, the mushrooms for emergency stops were provided for what is believed to be the normal working condition, with the contacts closed (non-emergency) then with their output variable (XOut) active, while for example the start buttons are provided to start with open contacts state, with the variable output not active. The user can set this starting condition by changing the value of the variable "Default output in simulation" in each Sensor Outputs tab.

By modifying these variables, it is possible to carry out simulations with very different machine start conditions and simulate the behavior of the module in case of failures (e.g. pushbuttons contacts stuck) of some devices.

The interaction of the Sensors through the blue areas are useful and practical for the general operation but in some cases, you might want to simulate non-standard operations or fault conditions detected by Sensors. For this reason, in simulation time, next to each sensor output variable, appear small buttons that allow you to set the values of the individual variables and to verify the behavior of the system under these conditions. Variables of type X have a small pushbutton with text "01", as they can take only one of these two values. Clicking on the button and then the corresponding output sensor continues to alternate between these two values.

Variables of type B and W have a small pushbutton with text "+-" and clicking on this button opens a small window where you can input the desired value.

Finally, the standard output fault XFault, presents a special button that can be activated only once and to cause some events (not visible) to the module.

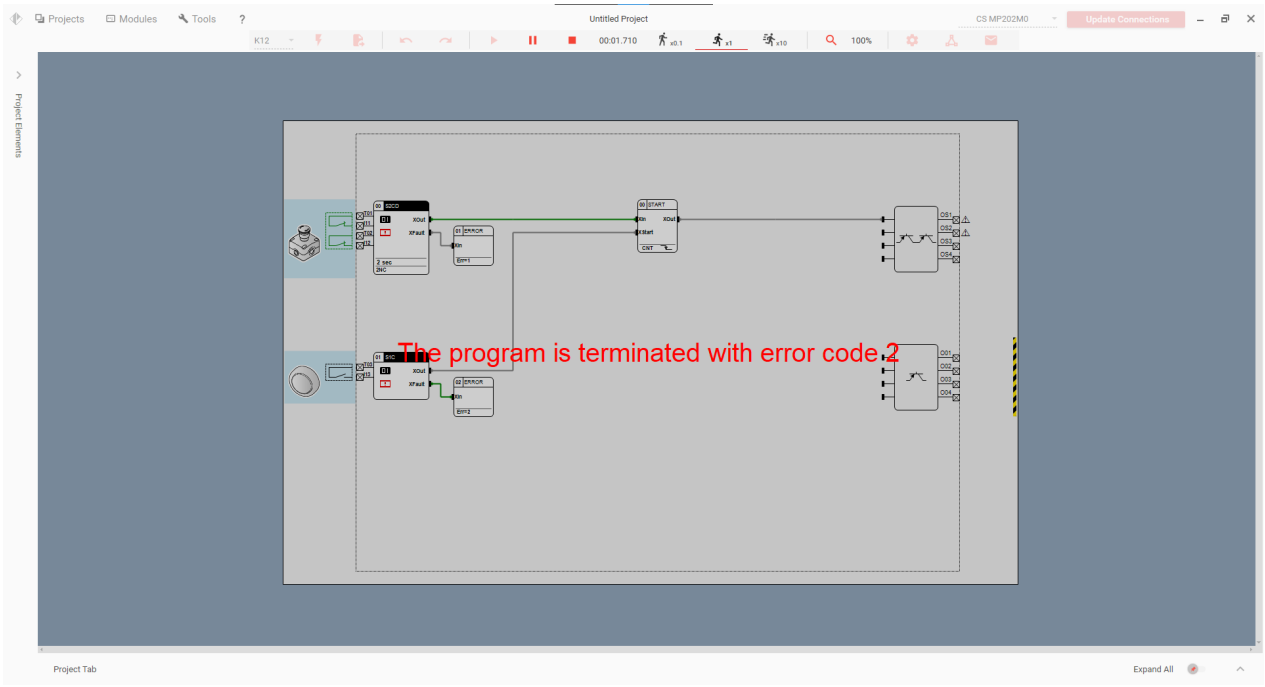
Please note. Through these buttons during the simulation, you can also enter values not foreseen by the current configuration. For example, in the case of a modal switch with 4 inputs, during simulation time is

 2024 © Pizzato Elettrica S.r.l.

possible to input a value greater than 4, although this actually will never happen. It has been preferred to leave this possibility to user in order to evaluate cases currently not foreseen.

5.7.7.3 Simulated ERROR conditions

In case the XFault variable (or any other variable) is linked to an ERROR functional block, the activation of the XFault will set the Error condition. In this case, the simulation is stopped at the instant where the first ERROR function is invoked and in the log box is highlighted an Error event (simulated) with its error code. You will also note that the outputs are "disconnected" even in cases where the control connector has value 1, while this does not happen for the signal outputs.



In reality, the module will set itself in exactly the same way in safe state, by turning off the safety outputs and keeping active signal outputs while the error code highlighting will be done via the two blue LEDs of processors (see also the ERROR functional block).

Recovery from the ERROR state condition is not possible. You can just restart the simulation from the beginning.

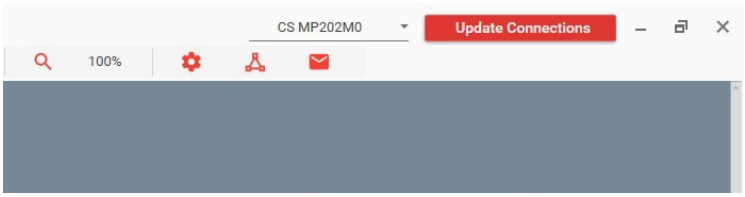
5.8 Monitor

You can monitor in real time the functioning of Gemnis module through the Monitor function, provided that the project file is in the default directory, the digital sign is the same on the project file and on the module and the project is not protected by the monitor password. It is necessary to start more instances of Gemnis Studio in order to monitor more than one module.

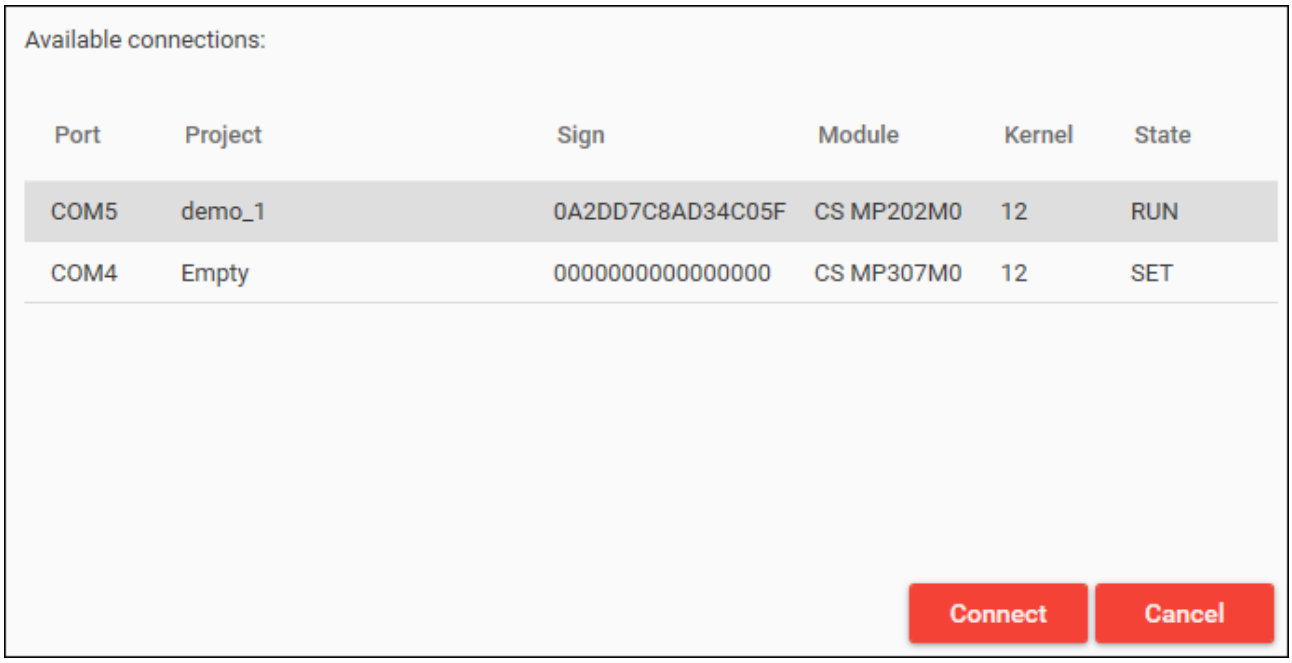
In case of password protection for the monitor function, it will be immediately request to enter the password.

In order to observe the operation of a module you need to connect it with the integrated USB port and then press Update Connections.

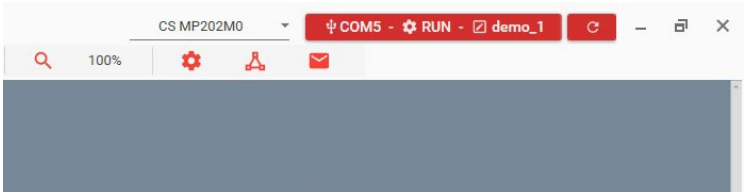
 2024 © Pizzato Elettrica S.r.l.



In case there is just one module plugged to the PC, Gemnis Studio will connect automatically. If there are two or more modules plugged at the same time the software will show the following window.



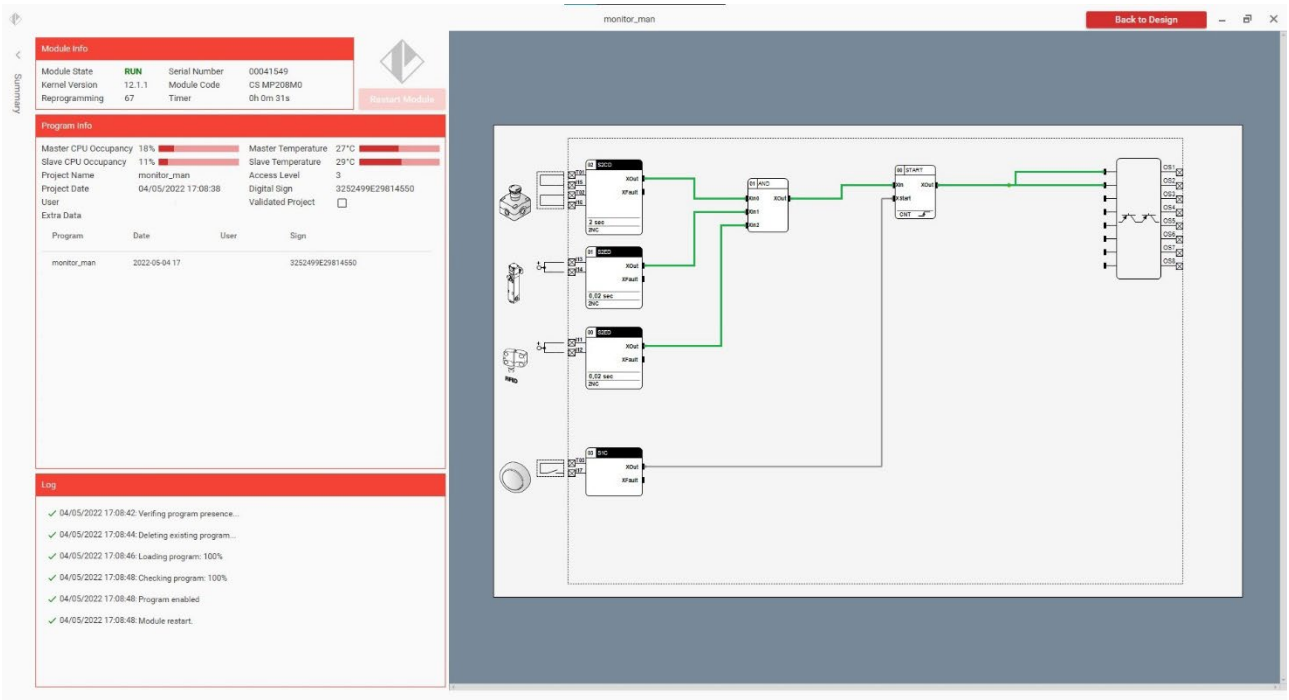
Once the module is connected, in the area of available connections you will see two buttons. The first contains the graphic symbol of the USB port and the COM port number (PC) associated, the status of the module, and the program name running into the module; the second one is a refresh button to update again the connections available.



The key for the update of the connections will be useful if more than one module is plugged to the PC.

Clicking on the first button, on top of the desktop, will opens the “Monitor”.

 2024 © Pizzato Elettrica S.r.l.



In the picture above, shows the monitor mode, with two areas: on the left is present the Summary with details of the module, of the program and the log of events, while on the right is present the real-time graphical visualization of the internal status of the module.

Summary

The summary includes the following details:

1. An area containing information relating to the module and to its state, in particular:
   - The status of execution of the module RUN, SET, or ERROR.
   - The version of Kernel.
   - The number of reprogramming carried out
   - Serial number of the module.
   - The code product of the module.
   - The value of the internal timer. Is reset at each switch and can be set by commands via USB (see System commands).
2. Area relative to the currently executing program or otherwise loaded into the module. In this area you can collect data of the main screen of the saved project:
   - Two indicators of the maximum time spent in each of the two processors (master and slave) to run the Application Program from the instant of the start of the module. The data is constantly updated, and is reset only by a restart of the module.
   - Two indicators related to the internal temperature of the module detected independently by the two processors.
   - The name of the running project.
   - The current access level set in the module.
   - The date of the Project (the date specified in the project file, not the date of loading in the module).

 2024 © Pizzato Elettrica S.r.l.

- The digital signature of the Application Program.
- The name of the user who created the project.
- Project validated: indicates whether the project executed by the module has been validated or not.
- The extra data user.

3. The list of the last 15 loaded programs
4. The log of events where are present messages from the program and other information during the programming phase. In case of ERROR in the log of events will be reported information regarding the problem.
5. In the form, there is also an icon of the module useful for identifying the connection status and a "Reboot" button that lets you restart the module directly from Gemnis Studio when the module is in SET or ERROR.

Gemnis Studio is capable of managing multiple monitors simultaneously with the only limitation of computer resources and screen space to display information. If a module is disconnected or reconnected while your Monitor tab is open, just listed data are deleted immediately. If the module is reconnected, you can update the data both by closing and reopening the monitor window, or by clicking with the mouse on the module symbol, which attempts an automatic reconnection to the module. In any case, be aware that the speed of reconnection is affected also the performance of the operating system of the computer.

Execution state control within the module

In monitor mode, you can see the graphical representation of the project, in a way very similar to what happens during the simulation, with the main difference that in this case you can’t interact but only observe the (real) operation of the module. The values of the inputs and outputs of the sensors, the functional blocks, and the outputs of the module are graphically drawn in exactly the same way as a simulation session.

Note: As the module does not have enough memory space to store the graphic descriptions of Application Program, in order to get access to the graphical display in the computer should be present the Project's PZZKE file. Because a project with the same name can be compiled and tested several times, to correctly show the module state, the question arises about, given a project name, what is the correct graphical representation.

Gemnis Studio show the graphical representation of an Application Program only if exist a Project file with the given name and the digital signs of both are the same.

To work around the problem of eventual overwriting of Application Program already sent to the module, Gemnis Studio allows you to save a Backup copy of the Project file for each transmission of Application Program to the module (see General options). So assuming for example that a program named "Test" has signature F1 and build date 20 July 2012 hours 14:20:00, at the time of submitting the program to the module (if the automatic backup option has been set) will create a file with name "Test-12-7-20-14-20-00.PZZKE" containing a copy of the Test program, in the backup directory. If the program is modified at 15:00:07 that the signature becomes F2, sending it to the module will create a file called "Test-12-7-20-15-00-07.PZZKE" and so on. When you connect to a module, and you require the viewing of its internal state, Gemnis Studio search in the default folder a Project file which has the same name as the and the same digital signature of the Application Program and, if that file exists, uses it for creating the graphical representation of the operation of the module. If this file does not exist, you can search in the backup file, the correct one. In case you want to restore a Backup file, to edit it you will need first to copy and rename the Backup file by deleting in the file name the part of the date. For example, in the case mentioned above you will have to copy one of the two files from the backup directory to your working directory and rename

 2024 © Pizzato Elettrica S.r.l.

the file to "Test.PZZKE".

ⓘ The digital signature is calculated from the data necessary for the correct operation of the module. These data do NOT include informative data as comments, notes or the graphical position of objects in the desktop. Therefore, if a program with signature F1 changes such as adding notes or moving functional blocks or connections, but keeping unchanged Sensors, Functional Blocks and Connectors, the digital signature of the file will always be F1 because the global function the module carried out is the same. Can then happen that two writings on the module may have the same digital signature. At this point if the user cancels the "cosmetic" changes and then reconnects to the module, the Gemnis Studio Monitor for graphical representation will be the last-saved version of a Project.

ⓘ Updating data in video does not happen in real time and update period is approximately 0.5 seconds (the speed of the computer on which you installed Gemnis Studio can stretch this time). Anyway, the shown data refer to a single cycle execution of the application program and are all temporally consistent.

5.9 Prints

From menu Projects → Prints are available the reports of Gemnis Studio.

Gemnis Studio is a multi-project environment and, as a result, if you have multiple projects open at the same time, reports will refer to the active project.

5.9.1 Validation Report

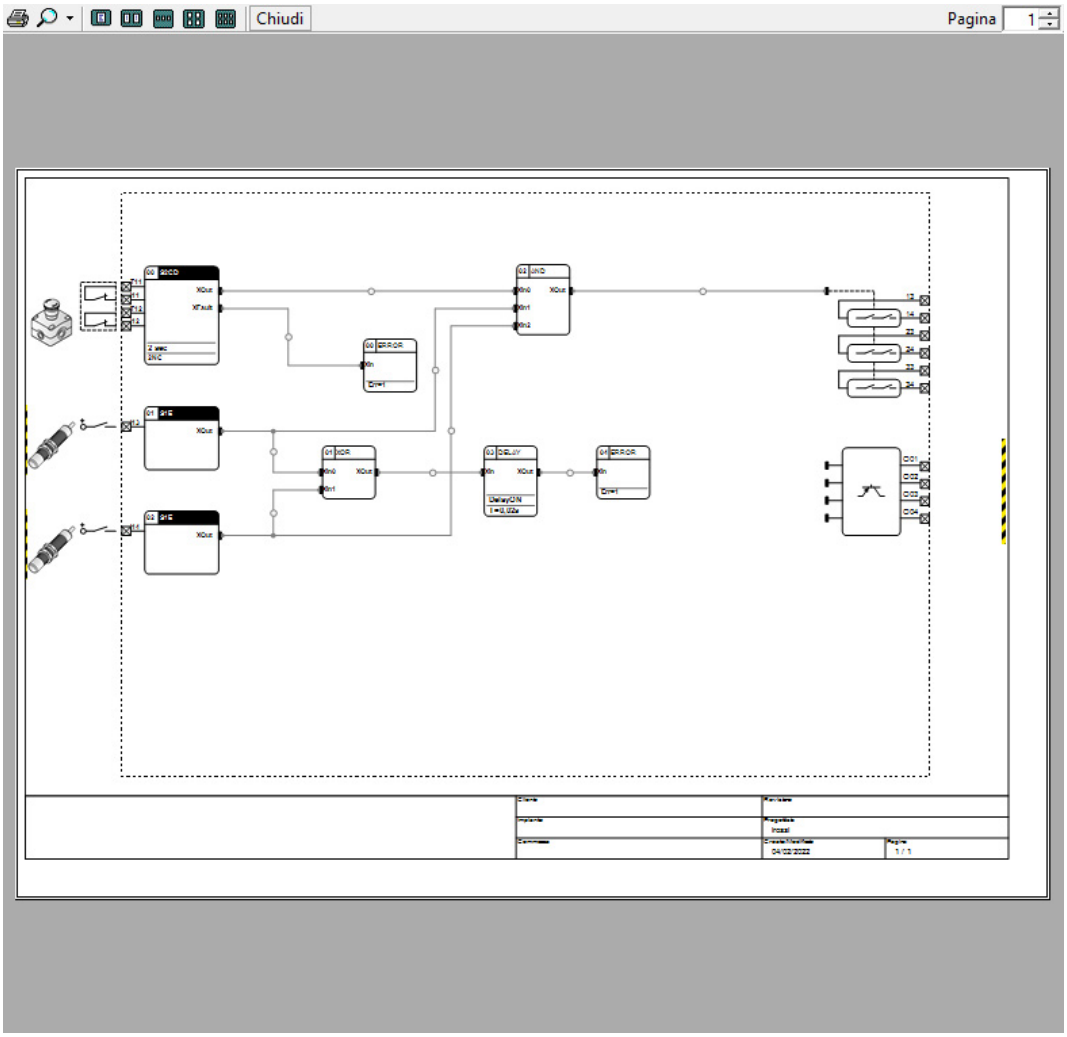
The validation report contains summary information of the entire project. The report contains information about:

- General Data of the project;
- Parameters of functional safety;
- Data on the sensors used;
- Formal evaluation;
- Front image of the module;
- Information on terminals;

 2024 © Pizzato Elettrica S.r.l.

5.9.2 User Program Report

The report of the user program is visible in the picture. This report is generated according to the "Desktop Settings" that define the size and orientation of Desktop pages so that there is a correlation between what you see on screen and what you print.



In the Title block appears the data set in the Tab "Project Data", “Title block data” (see 5.6.1).

 2024 © Pizzato Elettrica S.r.l.

6 System commands

6.1 Introduction

Once programmed a family Gemnis module is able to communicate and execute commands that comes from the USB port or any COM boards. These commands are called "System Commands".

Some commands can be executed only through the USB port, while others may be carried out both through the USB port. Finally, some commands can be set only by one of the two ports in order to avoid possible mistakes.

If you intend to send commands to the system through the USB port then the transmitting system must set the USB as a serial port emulation with the following parameters:

Speed: 9600 baud Parity: N

Data: 8 bit

Stop: 1 bit

Flow control: Xon/Xoff

6.1.1 System Commands format and limitations of use

The system commands have a very simple format, to use the minimum of resources (computation time, transmission bandwidth) of the module. They can be of two types:

- Run commands having format like "Pxx"Cr where xx is the code of the command and is the Cr character ASCII Carriage Return (0x0D) or the combination of Carriage Return and Line Feed (0x0D, 0x0A). "Quotes" should not be written to the command. To these commands the module responds with strings like "[Pxx =" followed by the requested data and terminated by the string "]Cr".
- Writing commands having format like "Wx"Cr where x is a string.

All responses provided by system commands begin with the character "[" and end with the character "]" followed by Carriage Return character (Cr). Because the Application Program also can send messages by functional block “Message”, the use of the characters "[" and "]" in all responses to the system commands allows distinguishing responses to system commands from messages generated by the Application Program.

ⓘ All numeric values in system commands are expressed in hexadecimal format.

Not all commands are executable by any user at any time. The program loaded into the module can be password protected to limit the possibility of interaction with the module. Some commands can be executed or less depending on the State (RUN, SET, ERROR) of the module.

⚠ Some commands are reserved for Gemnis Studio. The execution of arbitrary commands in SET state when the module is not protected by password may result in cancellation of the program included in themodule.

 2024 © Pizzato Elettrica S.r.l.

6.1.2 Commands executable in RUN state

The following commands can be invoked when the module is in RUN state

- P00 = base data = module state, current access level, eventual error code
- P01 = password setting to change access level
- P03 = fixed program data = Program Number, Version, Signature, Date of program
- P04 = fixed program extra data
- P06 = enable/disable USB messages
- P07 = invoke UserDump
- Writing Data commands (WD, W?, WK, WE)

6.1.3 Commands executable in SET state

The following commands can be invoked when the module is in SET state

- P00 = base data = module state, current access level, any error code
- P01 = password setting to change access level
- P03 = fixed program data = Program Number, Version, Signature, Date of program
- P04 = fixed program extra data

6.1.4 Commands executable in ERROR state

The following commands can be invoked when the module is in ERROR state but only if as a result of errors due to activation of a functional block ERROR or other external errors handled by the module. Are excluded then error States due to internal failures of hardware and software module.

- P00 = base data = module state, current access level, any error code
- P03 = fixed program data = Program Number, Version, Signature, Date of program
- P04 = fixed program extra data
- P07 = invoke UserDump

 2024 © Pizzato Elettrica S.r.l.

6.2 System Commands list

6.2.1 P00 Command

Command format: "P00"*Cr*

Recalls the base data of the module.

The response format is:

"[P00 = x, l, ee]"*Cr*

where:

- x = 1 if module state = RUN, 2 if state = SET, 3 if state = ERROR
- l = code of current access level, ranges from 0 to 3
- ee indicates the error code if the module state is ERROR, otherwise it is 0.

6.2.2 P01 Command

Command format: "P01 = xxxxxxxxxxxxxxxx*Cr*

Send the password x (16 ASCII characters) to access level 3. If the password is valid the access level 3 is set.

The response format is:

"[P01 = xx, n]"*Cr*

where:

- xx = “OK” if the password was found or “NO” if it is not found
- n = actual access level. If the password has not been found remains the previous level.

6.2.3 P03 Command

Command format: "P03"*Cr*

Fixed data of the Application Program. The response format is:

"[P03 = nnnn, vv, ffffffffffffffff, yyyymmddhhmmss]"*Cr*

where:

- nnnn = number of the program (only for pre-programmed devices). The data is in hexadecimal format, then for example “P100” is reported as “0064”.
- vv program version
- ffffffffffffffff indicates the digital signature of the file (16 ASCII characters)
- yymmddhhmmss = year, month, day, hour, minutes, and seconds. Date program (each two characters without hyphens or spaces between them), a total of 12 hexadecimal characters.

If the program is not loaded in memory these data are all zero.

 2024 © Pizzato Elettrica S.r.l.

6.2.4 P04 Command

Command format: "P04"*Cr*

Extra program data. These are data that Pizzato Elettrica or the customer want to associate with the device. For example, batches, user codes, etc.

The response format is:

"[P04 = pppppppppppppppp, uuuuuuuuuuuuuuuu, xxxxxxxxxxxxxxxxxxxxxxxx]"*Cr*

where:

- p = project name (16 ASCII characters)
- u = user name (16 ASCII characters)
- x = user data (24 ASCII characters)

6.2.5 P06 Command

Command format: "P06=x"*Cr*

Enable (x=1) or disable (x=0) the trasmission of user messages in the USB port.

The response format is:

- "[P06=1]"*Cr* if user messages have been enabled or
- "[P06=0]"*Cr* if not.

6.2.6 P07 Command (UserDump)

Command format: "P07"*Cr*

Request the UserDump that's the customized dump (depends on program to program) of a subset of the module data. At the time of receipt this command creates an instantaneous copy of the data on selected connectors in the program (see Connector controlled by User Dump) and then they are compacted into a 16 byte string and sent.

The response format is:

"[P07 = xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx]"*Cr*

where:

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx = 16 bytes of data, ASCII encoded (32 ASCII characters in total)

This command allows a run-time external control about the values of some module internal variables to process or replicate these values on PLC or synoptic panels. Whenever the external controller will perform a refresh of the module State, it sends the command "P07" and then waits for the response string to translate in the format of the controller and process it.

ⓘ Including brackets and *Cr* character the string has a length of 39 bytes that, at 9600 baud, are about 40 ms. Since the execution cycle of the application program amounts to 10 ms this implies that it is not possible to have a real-time control on controlled variables.

 2024 © Pizzato Elettrica S.r.l.

6.3 Writing data into the module

Gemnis series modules allow the possibility to send some data from the outside world to the module and to ensure that such data are evaluated by the Application Program. To do this requires the following conditions:

1. Who wrote the application program has expressly provided this opportunity by enabling a communication Sensor (via the USB port) to write data into the module by using protocol "Safe".
2. Who wrote the application program has created connections between output connection points of communication Sensor and the rest of the Application Program.
3. A particular protocol is used to write data in the module.

With regard to points 1 and 2, please refer to the chapter on the communication Sensor (see 5.7.2.11). Here it is recalled only that this Sensor allows the creation of 8 data (user-defined type) that are evaluated by the Application Program exactly how if they were output data from Sensors connected to physical devices.

The communication protocol involves 3 steps:

1. External controller (PC, PLC) writing data to a buffer in the module. This step can be repeated several times (maximum 7).
2. Read back by the external controller of the module buffer content plus a code to be used as confirmation code. This step allows the verification by external controller if values were written correctly in the buffer.
3. Confirmation by the external controller of the transfer of data from the module buffer to the Sensor outputs by sending the confirmation code or cancellation of the buffer in case the data is different from what was sent.

The process is transactional since it allows simultaneously data writing in multiple connection points of the communication Sensor.

The process of writing data in the module can be executed only from one port (USB or COM) in the project. The choice of the Sensor where to activate the "Safe" protocol corresponds to the choice of which port should be used.

⚠ To avoid the risk that the data to be write in the module may be modified during transmission it is required the full implementation of the Protocol described above. In particular, it is essential that the external system that controls the data to be written in the module verify the correctness of the data in the buffer (point 2 of the Protocol) before proceeding to the confirmation of the transaction.

6.3.1 WD Command

The command for writing data in the module buffer has the following structure:

"WDxx=yy"Cr for variables of type X or B or

"WDxx=yyyy"Cr for variables of type W.

where:

xx = the index of the variable. Values range is allowed from "00" to "07" where 00 is the first variable.

yy = variable value (in hexadecimal). The values are allowed:

"00" or "01" for variables of type X

 2024 © Pizzato Elettrica S.r.l.

from "00" to "FF" for variables of type B

from "0000" to "FFFF" for variables of type W

The write command execution does not gives any reply.

In practice this command allows to write in the module buffer that variable which index is xx must take the value yy.

The buffer capacity of the module is seven data so this command can be repeated up to 7 times before using a confirmation command or an erasing command. When the buffer is full, more buffer write commands will not be executed. If in the same transaction the same variable is assigned several times, the last value inserted will be used.

6.3.2 W? Command

This command is used to verify the data in the buffer and has the following structure:

"W?"*Cr*

This command has a response format like:

“[W? xx1=yy1,xx2=yy2, .. ,K=zzzz]”*Cr*

where:

" xx1=yy1,xx2=yy2, …, xx(n)=yy (n)" is the content of the intermediate buffer with data written via the WD command.

"zzzz" is the confirmation code, i.e. the data to send to the module using the WK command when it have been verified that the data in the buffer are correct and you want confirm their writing in the variables of the sensor.

6.3.3 WK Command

The transaction confirmation command has the following structure:

"WK=zzzz"Cr

where:

"zzzz" is the transaction confirmation code obtained from a command W?

This command response:

"[WK=OK]"Cr if the command was successful.

"[WK=NO]"Cr if the command goes wrong, for example because the transaction confirmation code is incorrect.

After execution of the WK command, module buffer is emptied both the command has been successful or not.

6.3.4 WE Command

The erasing of the intermediate buffer has the structure:

"WE"*Cr*

This command response:

"[WE = OK]"*Cr* if the command was received and executed.

This command typically is executed only when the data verification in the module buffer using the command W? shows that the buffered data is incorrect and it is therefore necessary to flush the buffer to

 2024 © Pizzato Elettrica S.r.l.

restart the whole process.

6.3.5 Example of transaction

Suppose you have created a project where there is a sensor in which the first three output variables have been defined as XOut0, BOut1, WOut2. The three variables have so index 0, 1 and 2 and are of type X, B and W.

Suppose you want to set these variables simultaneously, to decimal values: 1, 254 and 13743. The completion of the transaction is carried out through the following sequence:

Write command	Reply from module	Comment
WD00 = 01		Write command: value 1 in the 0 index variable, i.e. on XOut0.
WD01 = FE		The decimal value 254 shall be written in hexadecimal format i.e. FE
WD02 = 35AF	+	The decimal value 13743 must be written in hexadecimal format, i.e. 35AF
W?	[W? 00=01,01=FE,02=35AF,K=01EC]	The module responds with the contents of the buffer and the hexadecimal confirmation code 01EC.
WK = 01EC	[WK=OK]	At this time the command was executed, and the three values were written in the variables XOut0, BOut1 and WOut3.

 2024 © Pizzato Elettrica S.r.l.

7 Support

7.1 Website

Support for this family of products is given on-line from the web site www.gemnis.com where you can:

- Download the installation package from Gemnis Studio (after registration)
- Get the most up-to-date version of the manual (this document)
- Get the most current version of the file that contains the list of modules available for purchase
- Other support information that will be gradually added

7.2 Technical support

A technical support service is currently provided free for users who have registered on \the site and have activated Gemnis Studio through the activation procedure (see Demo mode and Standard mode). Please note that the procedure for activating the software needs only the physical link with a Gemnis series module.

Technical support is provided through:

- Via phone at + 039-0424-470930 in Italian and English
- Via mail to the address tech@pizzato.com in Italian and English

The required information must be relevant to the functionality of the module. Pizzato Elettrica doesn’t offer a consulting service based on the customer’s application.

Pizzato Elettrica reserves the right to change the terms of service at any time and at its indisputable discretion.

 2024 © Pizzato Elettrica S.r.l.

7.3 Making of pre-programmed modules, series CS MF

Pizzato Elettrica is able to provide a service of pre-programming Gemnis series modules. Upon customer's request Pizzato Elettrica can create module that leave the factory pre-programmed with a specific project. If the customer project does not need the USB communication port, it is also possible to provide a pre-programmed module without such port, for cost savings and to prevent product any type code modification.

The possibility of creating pre-programmed modules is under trade agreements (minimum quantities are required) and communication of the project by the customer.

The development process in this case would be as follows:

- The customer, with Gemnis Studio and a programmable module (e.g. CS MP202M0), develops and tests a project on a machine until it fully meets its needs.
- The customer sends to Pizzato Elettrica a copy of the project, with a signed copy of the document called “Supply contract pre-programmed module GEMNIS CS MF”.
- Pizzato Elettrica assigns a code to the project, for example "P150"
- If the customer project does not need USB port, Pizzato Elettrica will pre-program required modules with code "CS MF202M0-P150” (note the letter "F") without the USB port. In case the USB port is necessary in the code would become "CS MP202M0-P150”.
- Pizzato Elettrica send pre-programmed modules to the customer.

Pre-programming is suitable for modules that must be installed on machinery produced in series, where the software development process is stabilized and where the customer wants to minimize factory efforts (programming modules, software management) and to manage the module like a component, with a single product code.

 2024 © Pizzato Elettrica S.r.l.