INOVANCE

2021

Quick Start Guide

IT7000

INOVANCE TECHNOLOGY EUROPE

V1.1

TABLE OF CONTENTS

1 GENERAL DATA

Date: 21.10.2021

Hardware: IT7000

Software: InoTouchPad v0.8.8.20-R

Info: IT7000 Quick start guide

2 PURPOSE OF THIS DOCUMENT

The purpose of this document is to facilitate the start-up and programming of the IT7000 touch panel.

The document is divided into different sections where it is explained how to communicate with the screen and field devices such as PLC, drives, ..., and how to develop the different screens that make up the HMI project.

In order to use the InoToucPad software, you need an Ethernet cable or a mini usb communication cable.

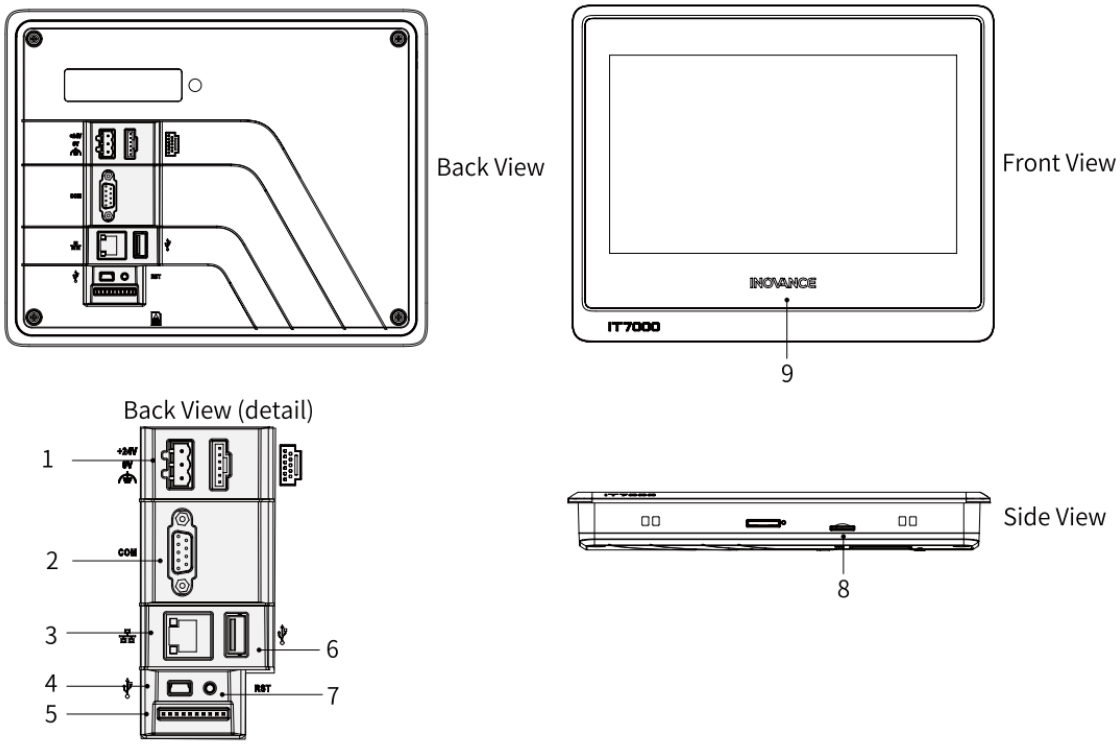
3 REVISION HISTORY

Revision	Date	Author	Description
1.0	27 May 21	RSR	First release
1.1	21 October 21	RSR	The following sections have been added: 9.2.2 MD Series communication 13 User Management 15 E-mail notifications configuration

4 WIRING & COMMUNICATION

4.1 TERMINAL DESCRIPTION

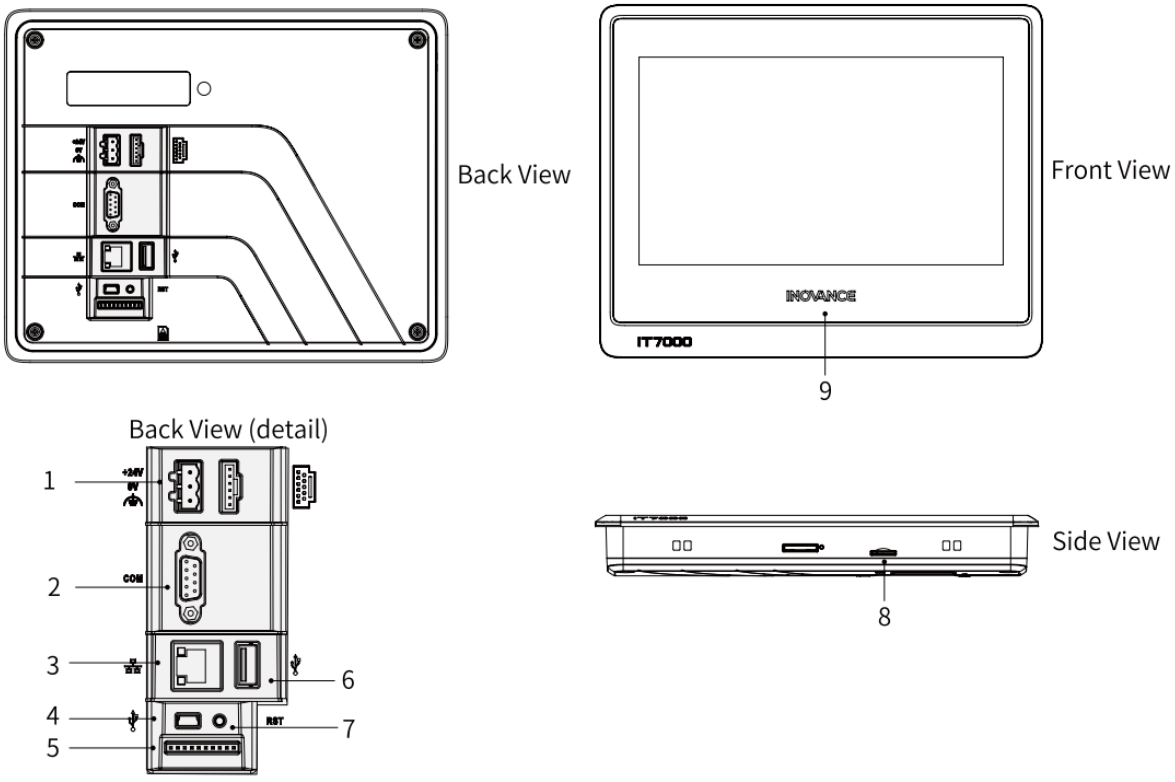
4.1.1 IT7070E



No.	Name	Function	No.	Name	Function
1	Power supply terminal	HMI 24 VDC power input terminal (A power connector is included in the delivery)	6	Mini USB port	The slave USB communication port to download and debug user programs in a PC
2	DB9 male connector	Communication port between HMI and peripherals Built-in COM1, COM2 and COM3 serial communication ports COM1: RS422 or RS485 COM2: RS232 COM3: RS485	7	Firmware upgrade port	For firmware upgrade
3	Ethernet port	Ethernet communication port (RJ45) for communicating with a PLC with a LAN port or a PC.	8	Micro SD card interface	Mounts a Micro SD card for data read/write and project file download
4	RESET button	Restores default settings	9	Power indicator	ON: normal Blinking: unstable OFF: power is off
5	USB type A port	The main USB communication port to read/write a USB drive and connect devices such as a			

		mouse or a printer			

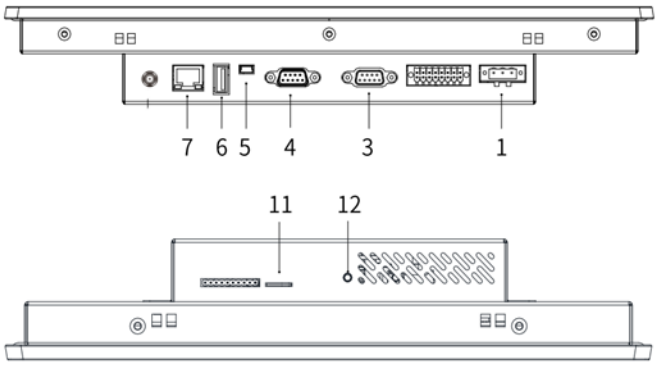
4.1.2 IT7100E



No.	Name	Function	No.	Name	Function
1	Power supply terminal	HMI 24 VDC power input terminal (A power connector is included in the delivery)	6	USB type A port	The main USB communication port to read/write a USB drive and connect devices such as a mouse or a printer
2	DB9 male connector	Communication port between HMI and peripherals Built-in COM1, COM2 and COM3 serial ports COM1: RS422 or RS485 COM2: RS232 COM3: RS485	7	RESET button	Restores default settings
3	Ethernet port	Ethernet communication port (RJ45) for communicating with a PLC with a LAN port or a PC.	8	Micro SD card interface	Mounts a Micro SD card for data read/write and project file download
4	Mini USB port	The slave USB communication port to download and debug user programs in a PC	9	Power indicator	ON: normal Blinking: unstable OFF: power is off
5	Firmware upgrade port	For firmware upgrade			

Connection between DB9 Male Connector and External Devices

4.1.3 IT7150E

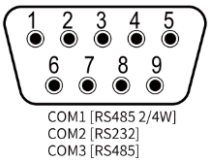


Front view

No.	Name	Function	No.	Name	Function
1	Power supply ports	24 VDC power input ports of HMI, which are +24V and GND (The power supply terminal used to connect power supply ports is delivered by default.)	7	Ethernet port	Ethernet communication port (RJ45) used to access a PC or PLC with LAN port
3	DB9 female	Communication port between HMI and PLC, built-in with serial communication ports COM1 (RS485/RS422) and COM3 (RS485)	9	Power supply indicator	Solid ON: power supply ON Blinking: power supply unstable Solid OFF: power supply OFF
4	DB9 male	Communication port between HMI and PLC, equipped with serial communication port COM2 (RS232)	10	Network status indicator	Blinking: communicating Solid OFF: no communication
5	Mini USB port	Slave port of USB communication used for user program downloading/commissioning through PC	11	SD card interface	Connected to Micro SD card, used for upgrading the firmware or kernel
6	USB type-A port	Main port of USB communication used to read and write the data in the U-disk, which can be connected to devices such as a mouse and printer	12	RESET key	Used to restore to default settings

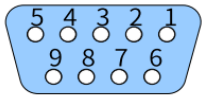
4.1.4 IT7070E/IT7100E DB9 PORT

The DB9 male connector has three built-in serial communication ports COM1, COM2 and COM3. The pins are described in the following table:

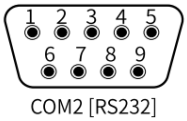
Pin No.	COM1[RS485] 2 Wires	COM1[RS485] 4 Wires	COM2 [RS232]	COM3 [RS485]	DB9 male
1				RS485-	
2			RS232 RXD (data reception)		
3			RS232 TXD (data transmission)		
4		TX+ (transmission positive)			
5	GND (signal ground)				
6				RS485+	
7	RS485-	RX- (reception negative)			
8	RS485+	RX+ (reception positive)			
9		TX- (transmission negative)			

4.1.5 IT7150E DB9 PORT

The DB9 female is built-in with two serial communication ports, COM1 and COM3. The pins are described in the following table:

Pin No.	COM1[RS485] 2 Wires	COM1[RS485] 4 Wires	COM3 [RS485]	DB9 Female
1			RS485-	
2				
3				
4		TX+ (transmission positive)		
5	GND (signal ground)			
6			RS485+	
7	RS485-	RX- (reception negative)		
8	RS485+	RX+ (reception positive)		
9		TX- (transmission negative)		

The DB9 male is built-in with COM2 communication port (RS232) to connect the controller with RS232 communication port. The pin assignment is as follows.

Pin No.	COM2 [RS232]	DB9 male
1		
2	RS232 RXD (data reception)	
3	RS232 TXD (data transmission)	
4		
5	GND (signal ground)	
6 to 9		

4.2 COMMUNICATION

The programming software for the IT7000 screen is the InoTouchPad. With this software it is possible to develop a project for the IT7000 screen and load it through the USB or Ethernet interface.

USB Port

Mini USB: used to connect the PC with a universal USB communication cable, to upload/download user configuration program and set HMI system parameters.

To connect through the USB port you need a cable with a MiniUSB connector. To log in to the HMI through USB, connect the MiniUSB port.

Type A: used to connect a USB drive, USB mouse or USB keyboard, plug and play.

Ethernet Connection

The 10M/100M adaptive Ethernet port, located on the back of the equipment, can be used for:

1. HMI configuration upload/download, system parameter setting and on-line configuration simulation.

2. Connection with multiple HMIs to achieve multi-HMI online communication.

3. Communication with the PLC.

4. Connection with a HUB or Ethernet switch through a standard Ethernet cable to join a LAN, or connection with the Ethernet port of a PC through a crossover

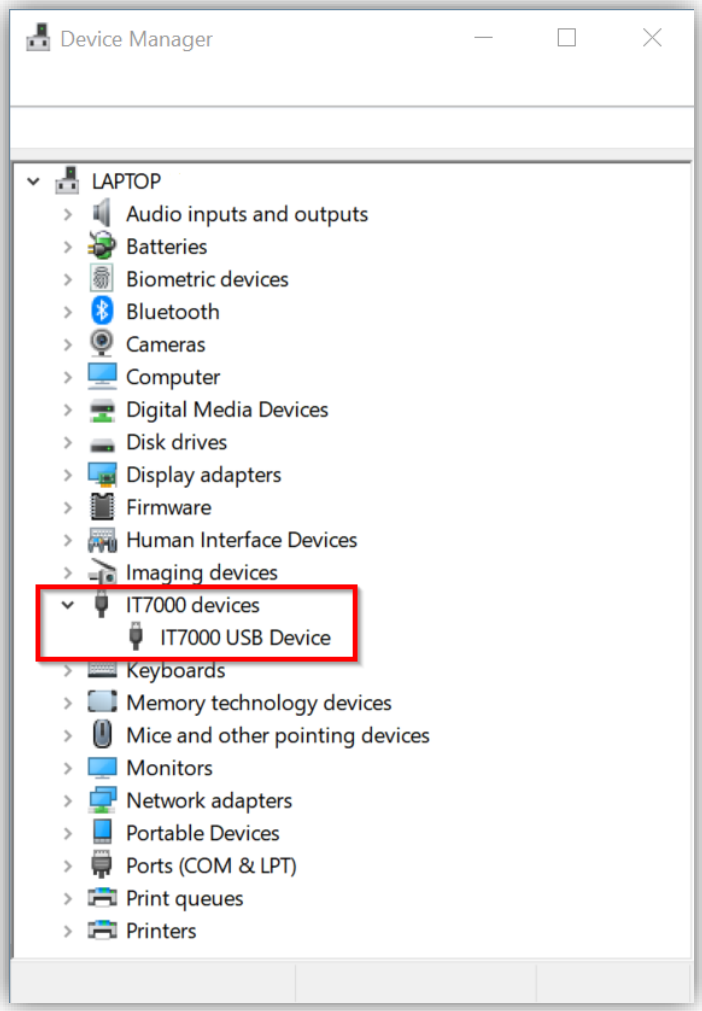
cable.

NOTE Use a shielded cable to ensure stable communication.

4.2.1 COMMUNICATION THROUGH USB

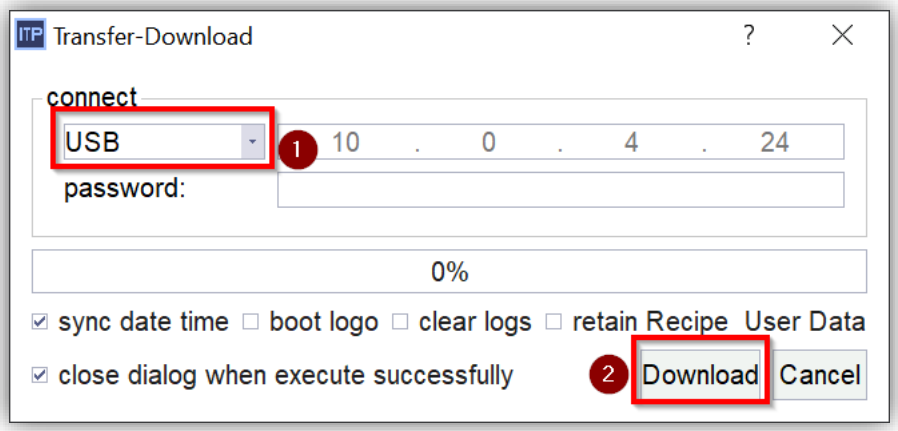
After installing the In TouchPad software connect the computer to the display through the mini USB port.

The software installer has already installed the necessary drivers so that the computer can connect via USB to the screen. The following image shows how the operating system has detected the IT7000 screen:



When the screen is connected, the project can be downloaded/uploaded by clicking on the download/update icons  .

The following screen is displayed to download the project. Select the USB connection and click on the download button:

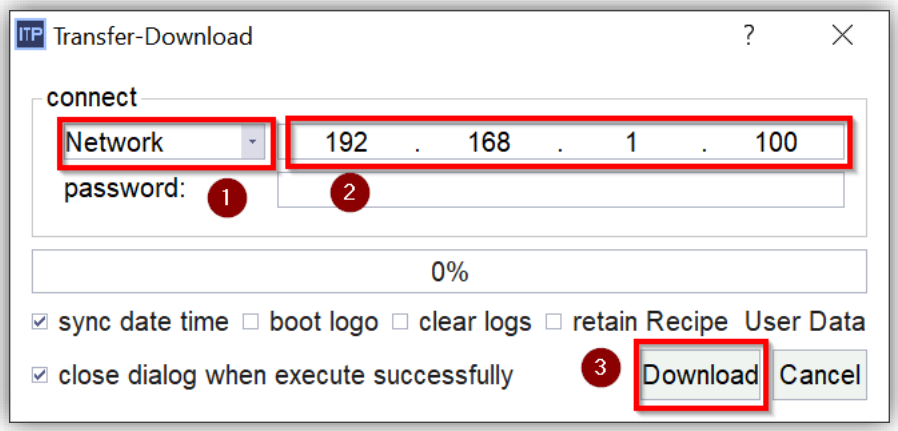


4.2.2 COMMUNICATION THROUGH ETHERNET

The InoTouchPad software can also communicate with the IT7000 display through an Ethernet connection. To know the IP address of the screen, access the system menu as explained in the section 5.2 Change IP Address.

When the screen is connected, the project can be downloaded/uploaded by clicking on the download/update icons  .

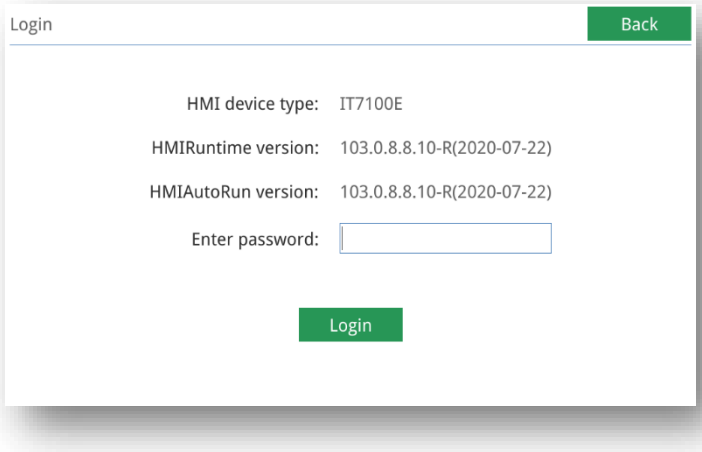
The following screen is displayed to download the project. Select the Ethernet connection, set the IT7000 IP address and click on the download button:



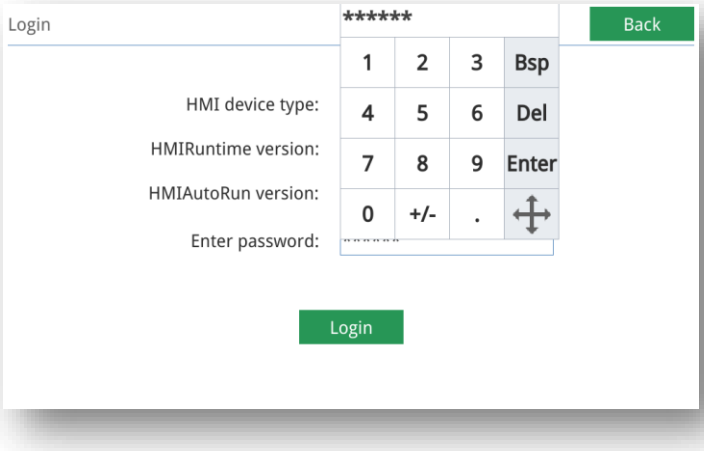
5 SYSTEM SETTING MENU

To enter the system configuration screen, follow the next steps:

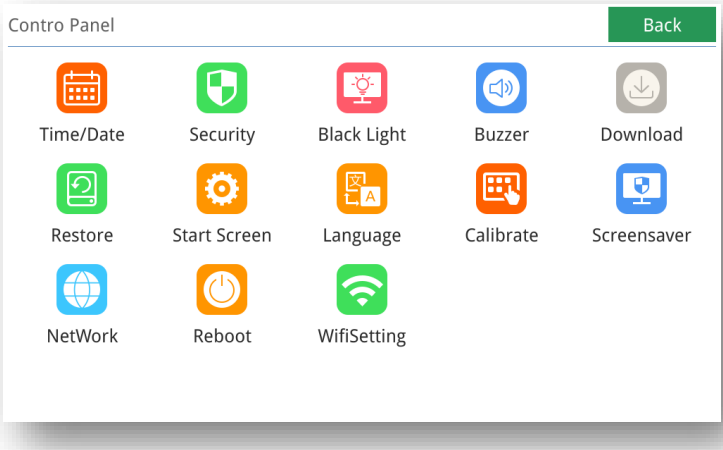
1. Power off touch screen
2. Keep pressing the screen and power on touch screen until the “password” window below appears



3. Enter the default password: 111111

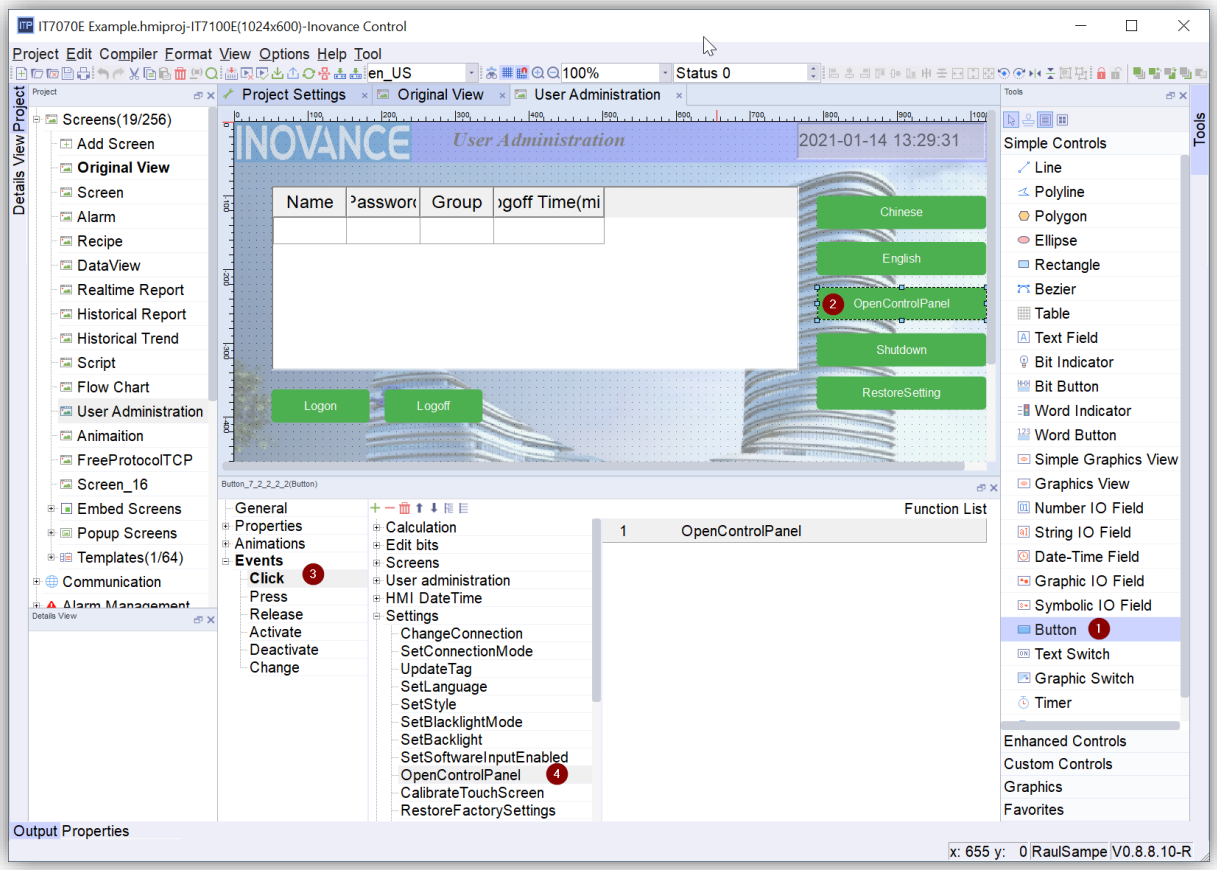


4. The system configuration screen appears:

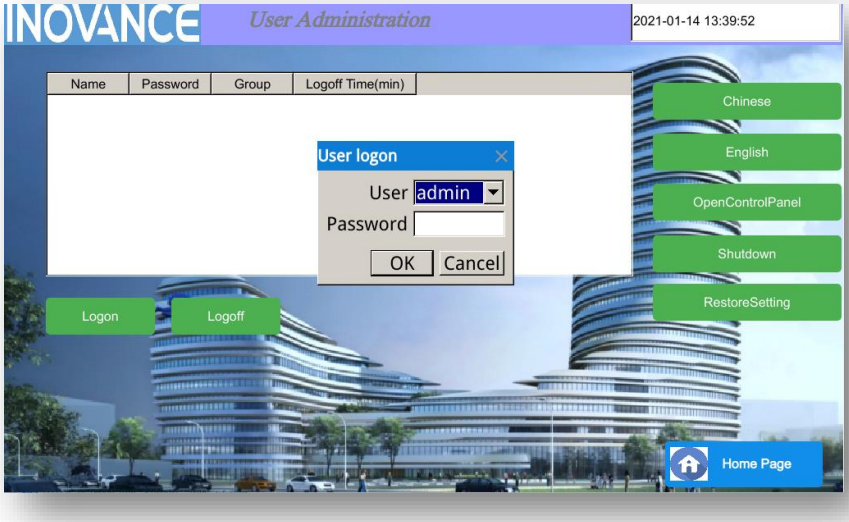


5.1 ACCESS SYSTEM SETTINGS THROUGH HMI SCREEN

IT7000 provide another convenient way(recommend) to get into setting view, using system function 'OpenControlPanel'. With this function, users can access to setting view at any time without power off-on the HMI.

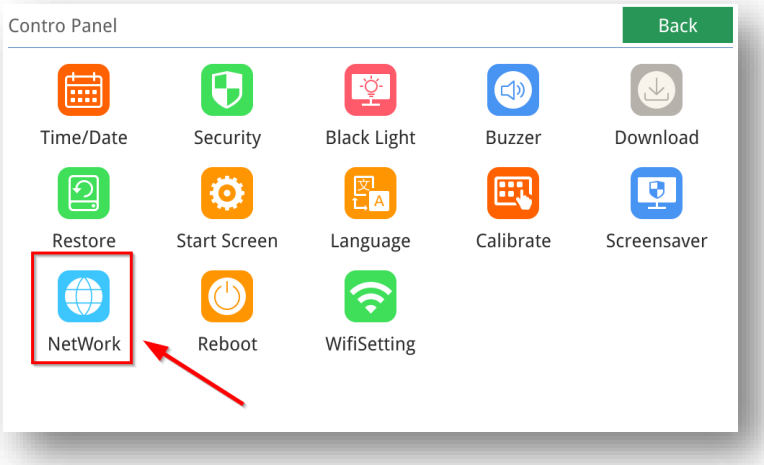


1. Select button tool
2. Insert a button in screen
3. Open button events, and select click event
4. Assign "OpenControlPanel" to click event
5. This is the result when click on OpenControlPanel button

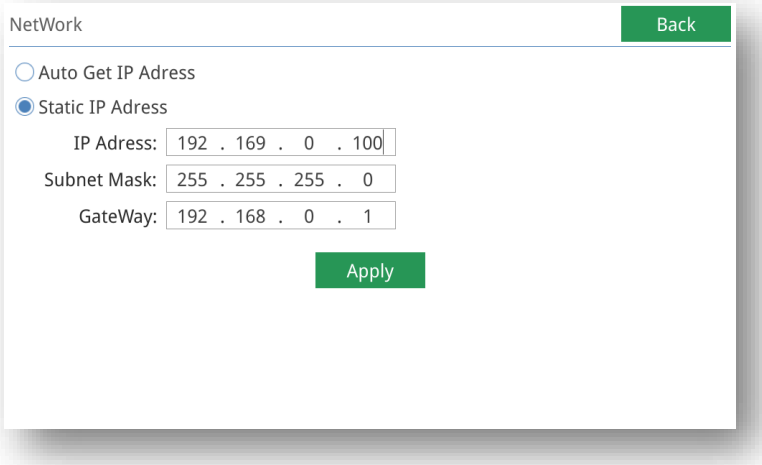

INOVANCE
User Administration
2021-01-14 13:39:52
Name
Password
Group
Logoff Time(min)
User logon
User admin
Password
OK
Cancel
Chinese
English
OpenControlPanel
Shutdown
RestoreSetting
Logon
Logoff
Home Page

5.2 CHANGE IP ADDRESS

Through system settings screen enter to network settings

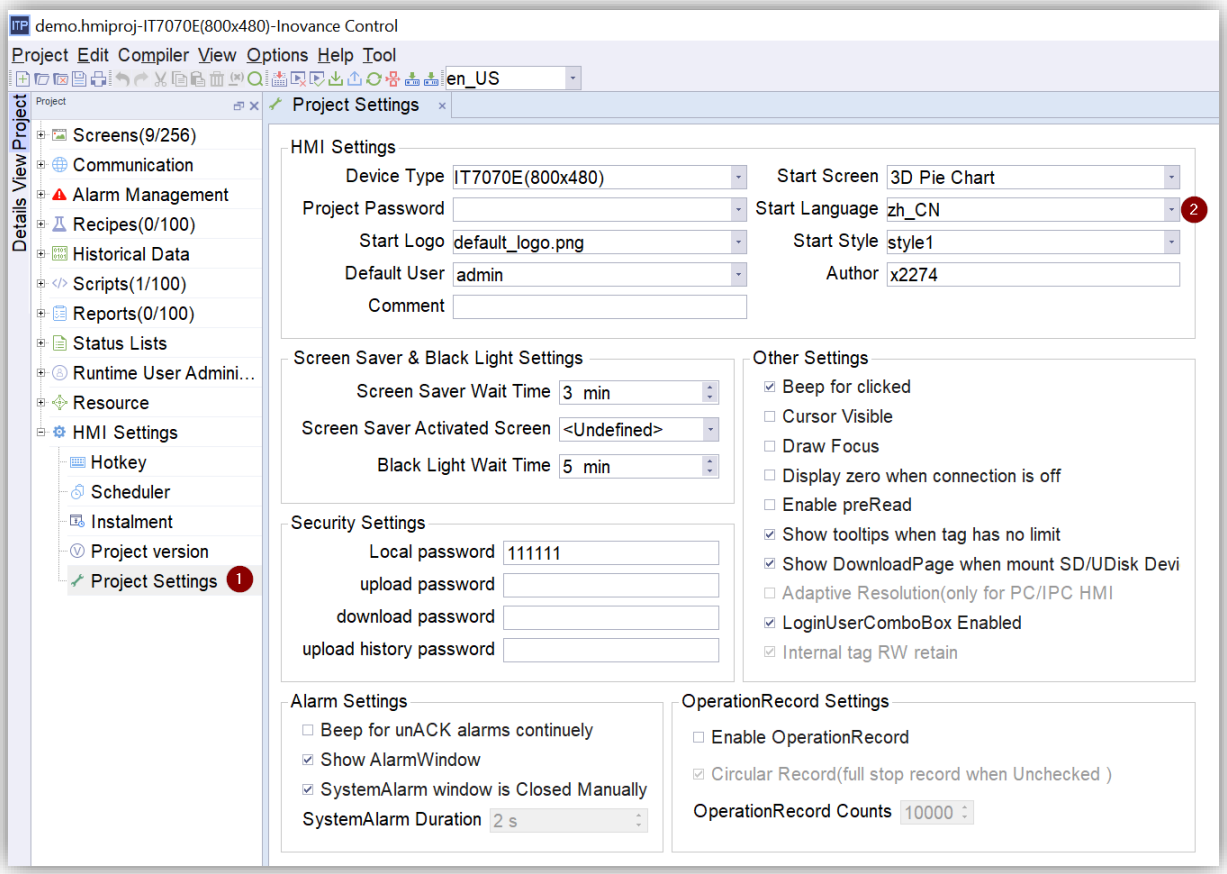


Change the network settings:

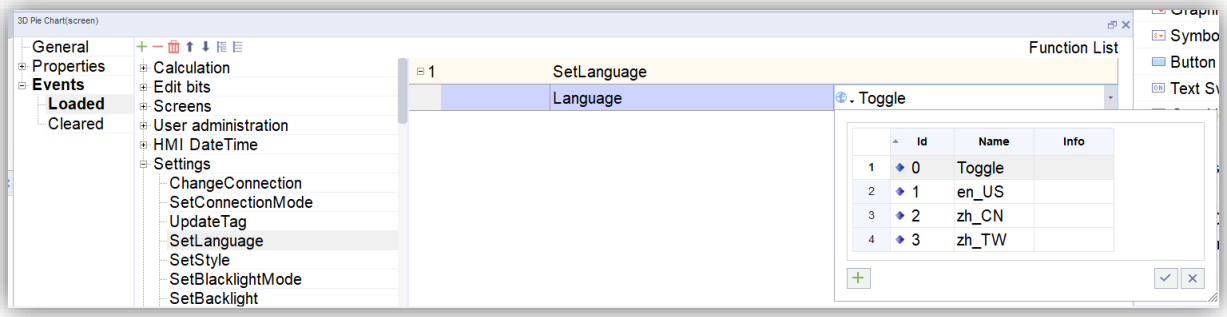


5.3 CHANGE PROJECT LANGUAGE

To change the language with which the HMI project is started, access the configuration screen and in the "HMI Settings" section modify the initial language.



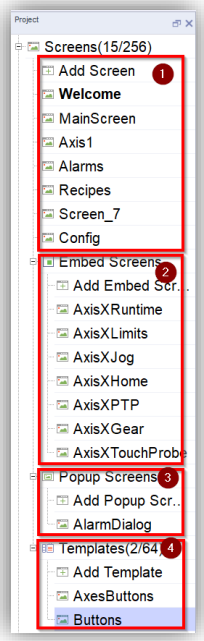
The language of the texts displayed on the screen can be changed from the "Runtime" of the screen using an button event:

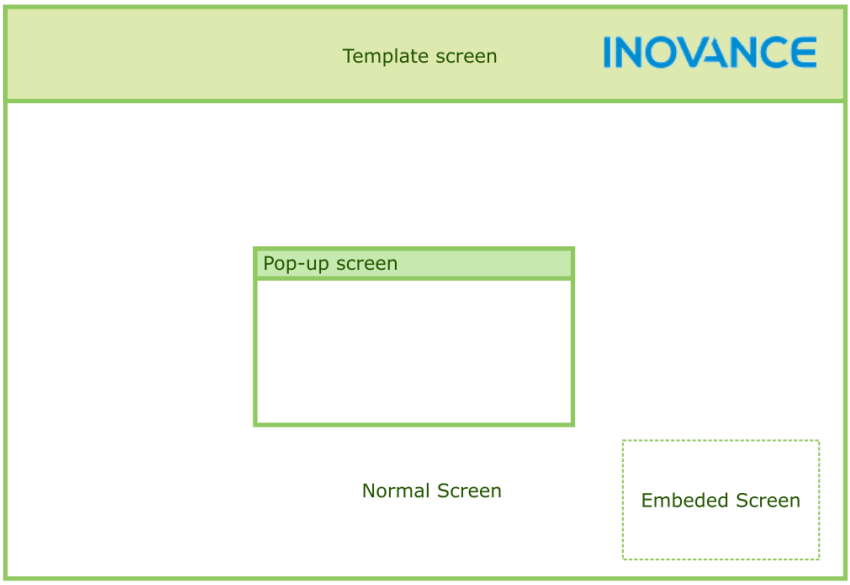


6 SCREENS

As the most basic element of the configuration, "screen" is used to present the content of the configuration. By creating a page, the operator can easily control and monitor the machine process and data of the device. The screen includes normal screen, pop-up screen, template screen and embed screen.

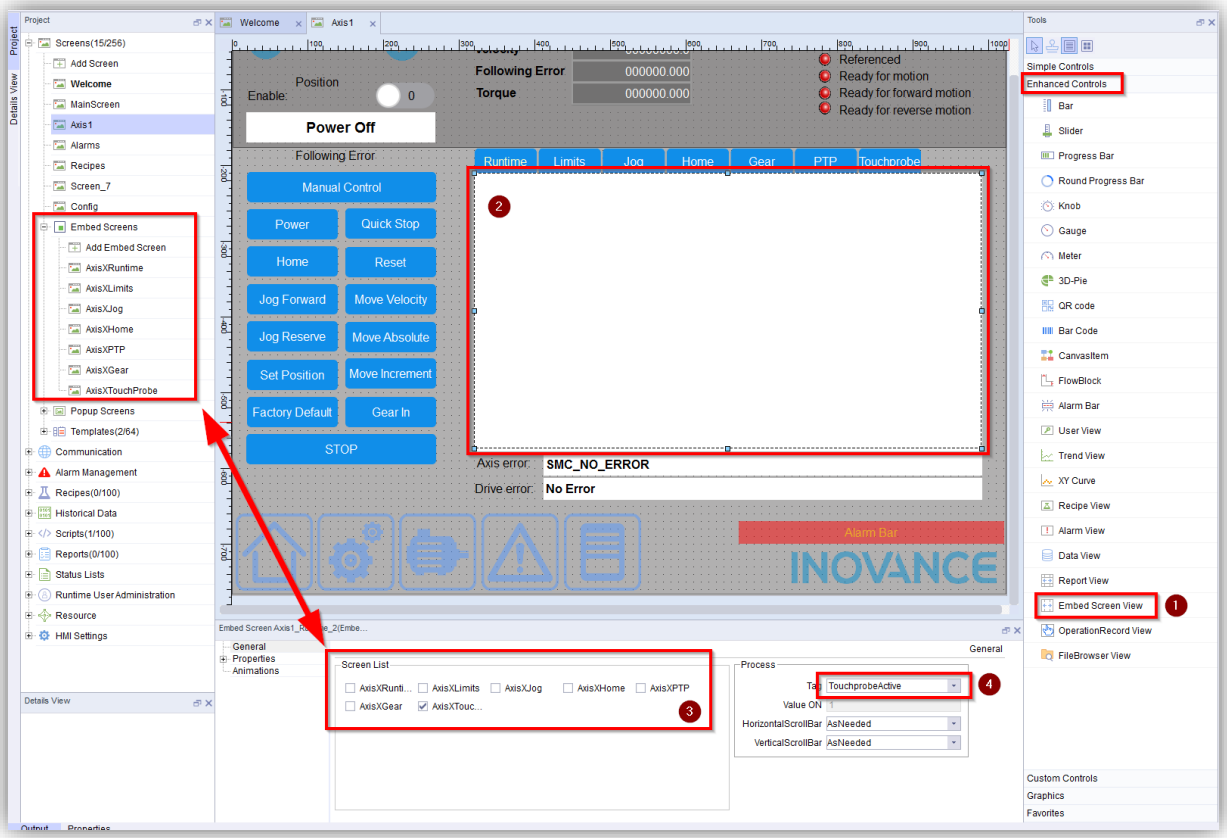
1. Normal screen: the main screen displayed
2. Embed screen: It is a screen inserted or embedded within another screen
3. Pop-up screen: can be used to display information above the main screen
4. Template screen: used as the basis for other normal screens to display controls common to more than one screen



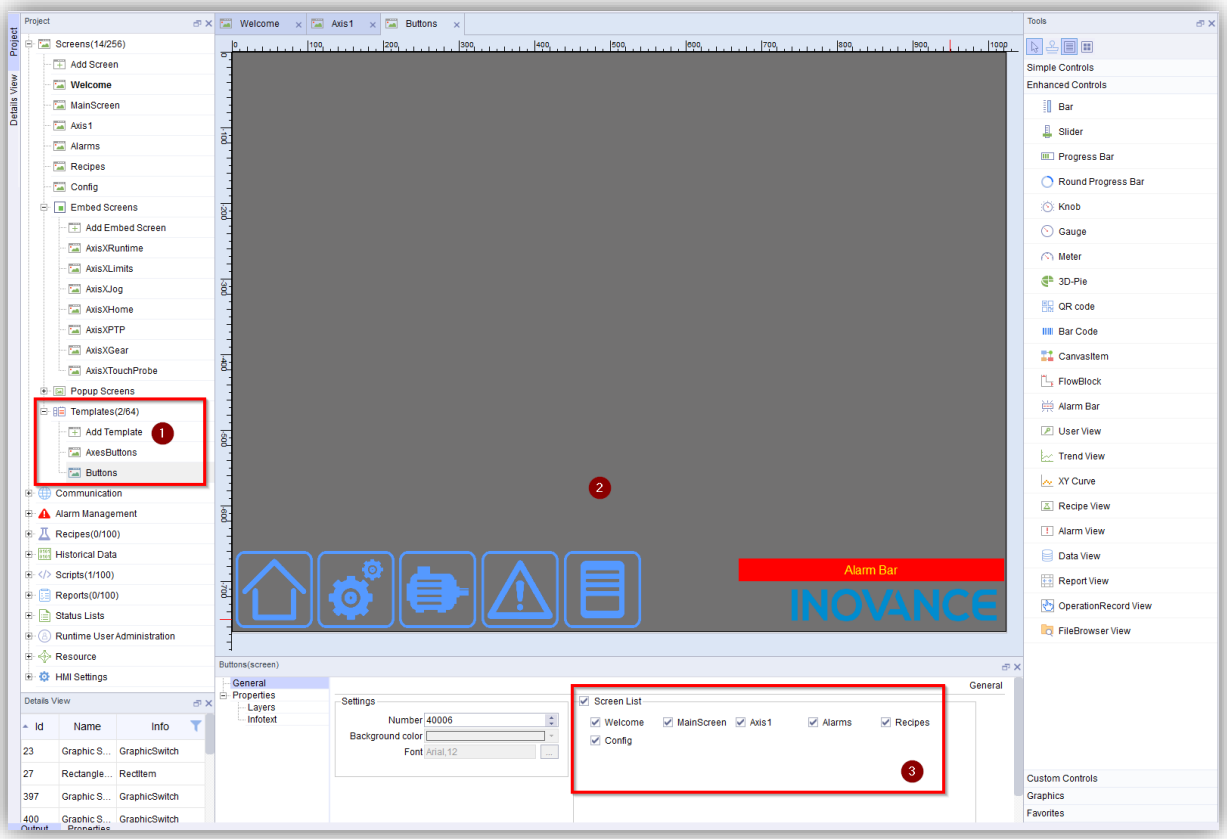


The following image shows the different types of views that can be closed in the HMI application:

6.1 EMBEDDED SCREEN



6.2 TEMPLATE SCREEN

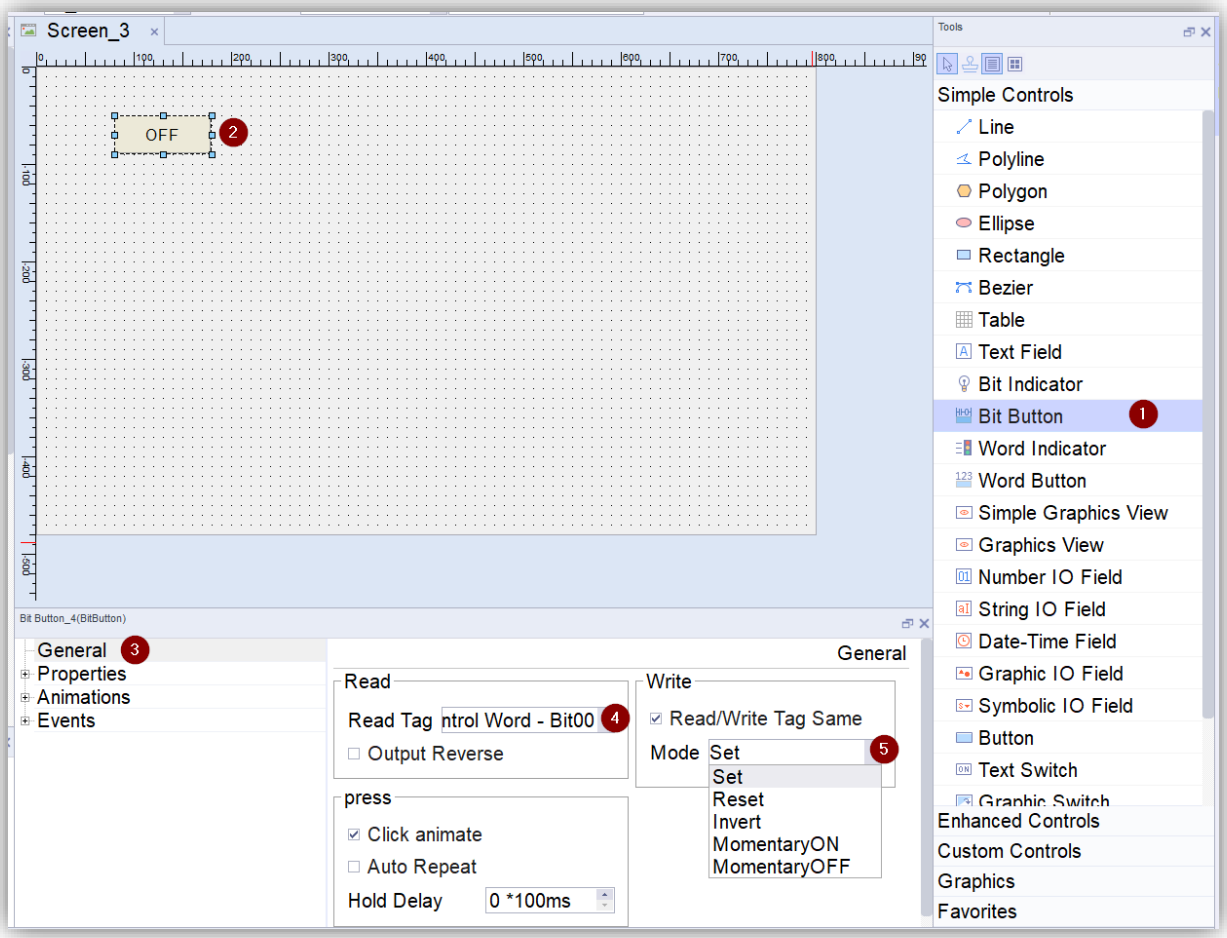

Project
Welcome
Axis1
Buttons
Screens(14/256)
Add Screen
Welcome
MainScreen
Axis1
Alarms
Recipes
Config
Embed Screens
Add Embed Screen
AxisXRuntime
AxisXLimits
AxisXJog
AxisXHome
AxisXPTP
AxisXGear
AxisXTouchProbe
Popup Screens
Templates(2/64)
Add Template
AxesButtons
Buttons
Communication
Alarm Management
Recipes(0/100)
Historical Data
Scripts(1/100)
Reports(0/100)
Status Lists
Runtime User Administration
Resource
HMI Settings
1
2
3
Alarm Bar
INOVANCE
Tools
Simple Controls
Enhanced Controls
Bar
Slider
Progress Bar
Round Progress Bar
Knob
Gauge
Meter
3D-Pie
QR code
Bar Code
CanvasItem
FlowBlock
Alarm Bar
User View
Trend View
XY Curve
Recipe View
Alarm View
Data View
Report View
OperationRecord View
FileBrowser View
Custom Controls
Graphics
Favorites
Details View
Id
Name
Info
23
Graphic S...
GraphicSwitch
27
Rectangle...
RectItem
397
Graphic S...
GraphicSwitch
400
Graphic S...
GraphicSwitch
Buttons(screen)
General
Properties
Layers
Infotext
Settings
Number 40006
Background color
Font Arial,12
Screen List
Welcome
MainScreen
Axis1
Alarms
Recipes
Config
General

7 QUICK EXAMPLES

7.1 BUTTON TO ENABLE DISABLE A BIT

This is the simplest procedure to activate a bit of the PLC or internal memory of the HMI with a button:

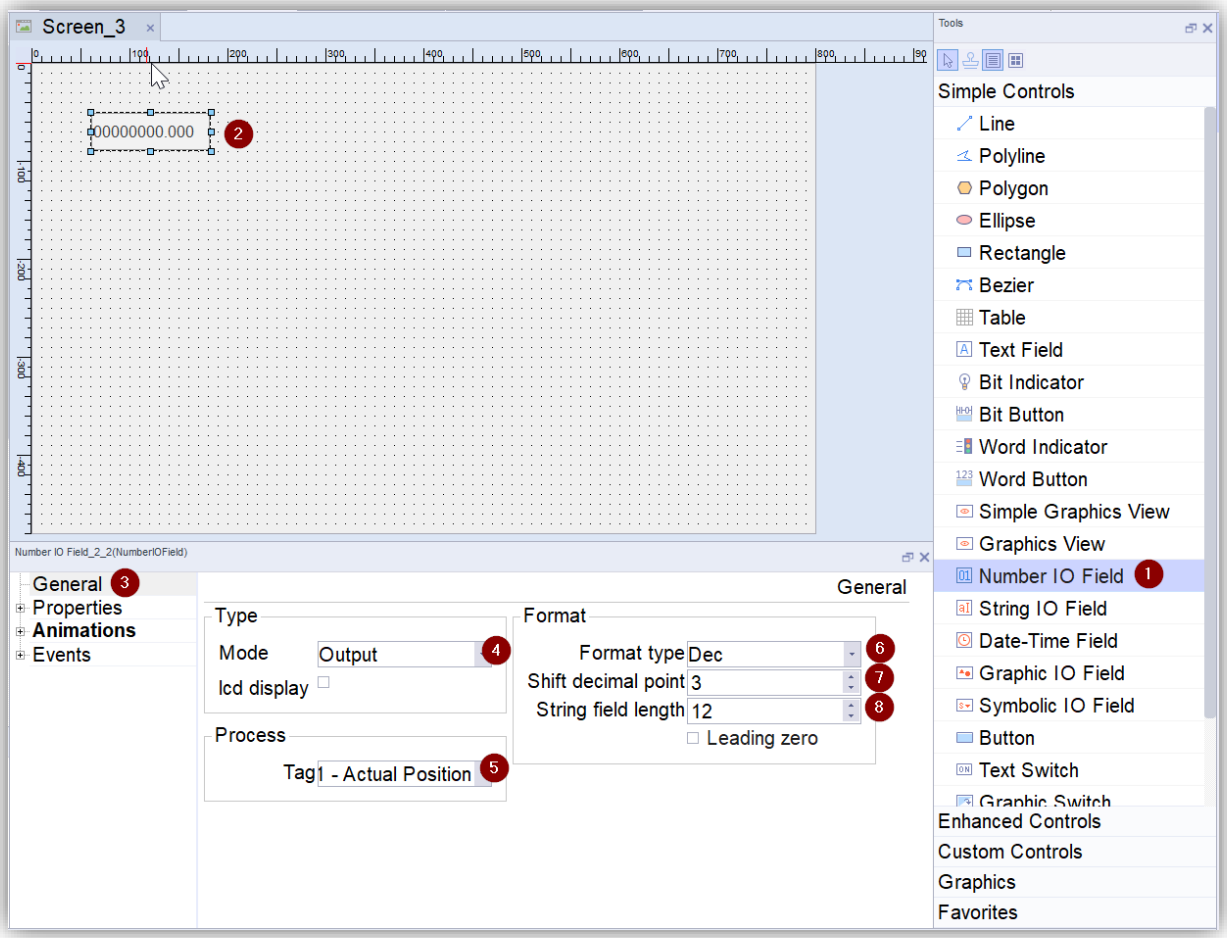
1. Select the "Bit button" component.
2. Insert the component into the screen layout.
3. Access the general properties of the component.
4. Select the bit-type variable (TAG) that will be activated by pressing the button.
5. Assign the action that the button will perform.



7.2 EDITBOX TO READ/WRITE VALUE

This example shows how we can read / write the value of a variable through the "Number IO Field" component.

1. Select the " Number I / O Field " component.
2. Insert the component into the screen layout.
3. Access the general properties of the component.
4. Configure if the component will be read-only or read-write.
5. Select the variable (TAG) that we want to read / write.
6. Set the data format: Hex, Dec, Bin, BCD.
7. Assign the number of decimal places.
8. Configure the number of digits the component will display.

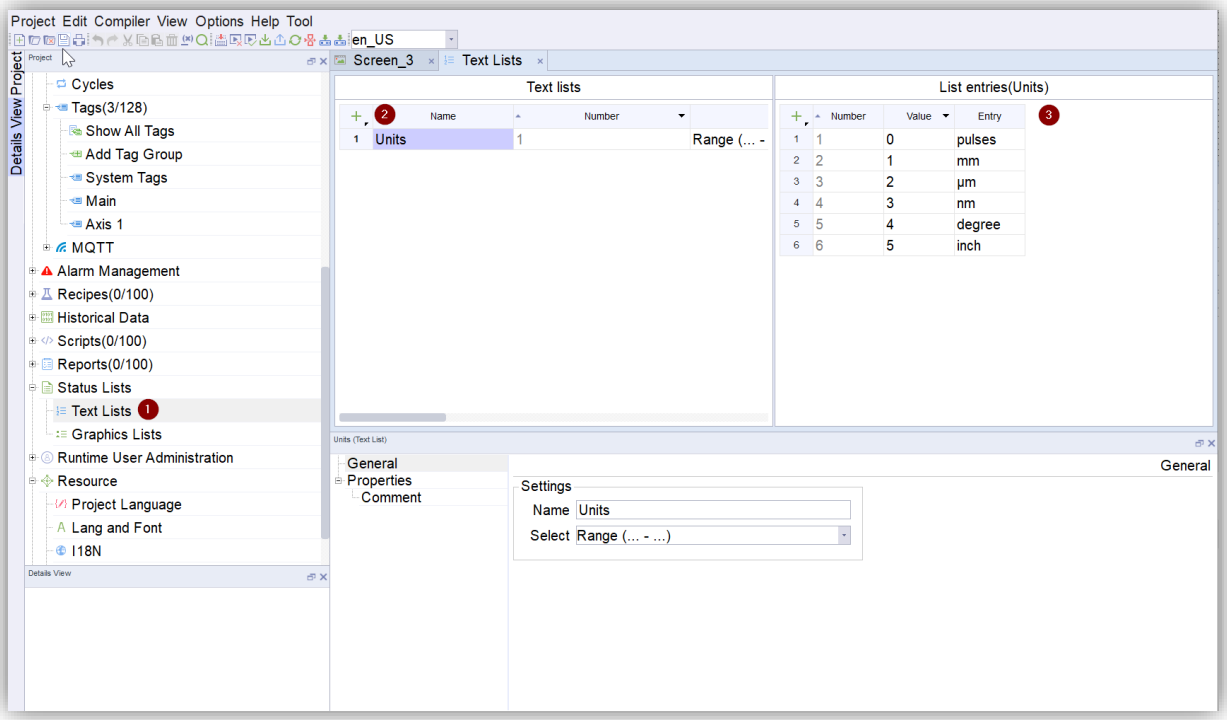


7.3 TEXT LIST

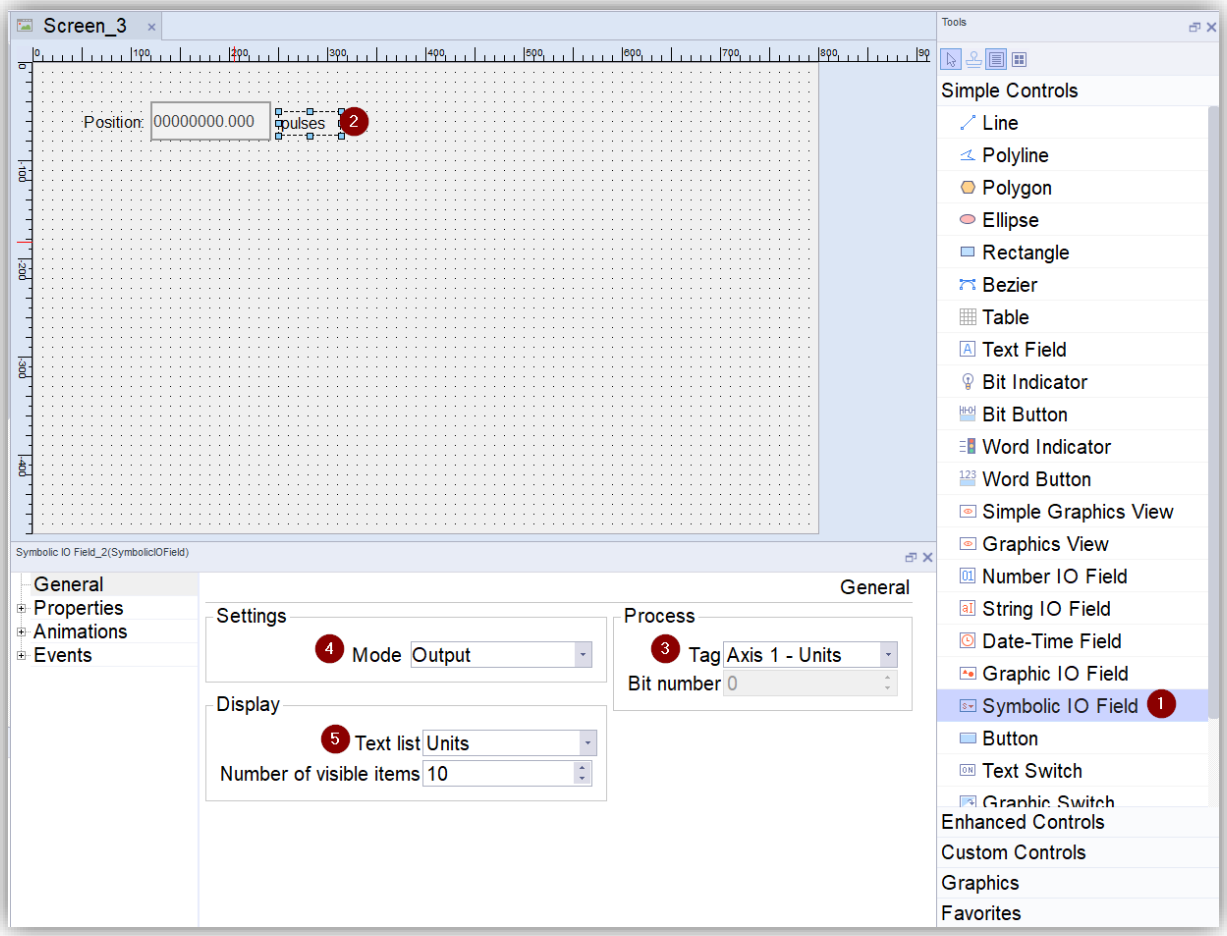
If we want to show different texts in the same component we will use the "Symbolic IO Field". This component is useful, for example, when we want to show the status of an element. It allows us to change the displayed text according to the value of a variable.

The following example shows how to configure a label that shows the configured length units (pulses, mm, ...):

1. First it is necessary to create a text list with the texts that we want to show. We access the text list editor from the project tree
2. Add a new list of texts
3. Add the texts that we want to show with a numeric identifier for each text. This identifier will be the one we use to change the text according to the value of a PLC or HMI variable.



1. Select the "Symbolic IO Field " component.
2. Insert the component into the screen layout.
3. Select the variable that indicates the type of length units selected
4. Set this component as read-only
5. Select the text list that we have created before with the texts of the units

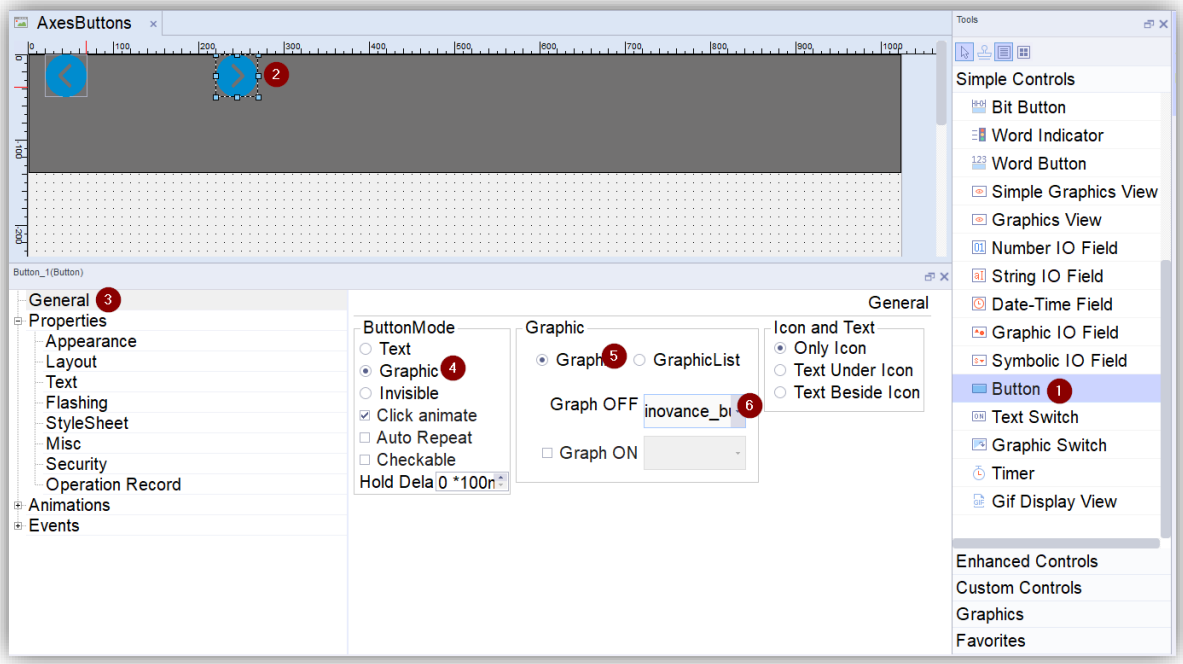


7.4 BUTTON WITH IMAGE

There are two components that allow us to create a pushbutton with one or more images. The "Button" component allows you to configure one or more images, and the "Simple Graphics View" component only allows you to configure one image for the ON state and another for the OFF state. Two examples with these components are described below.

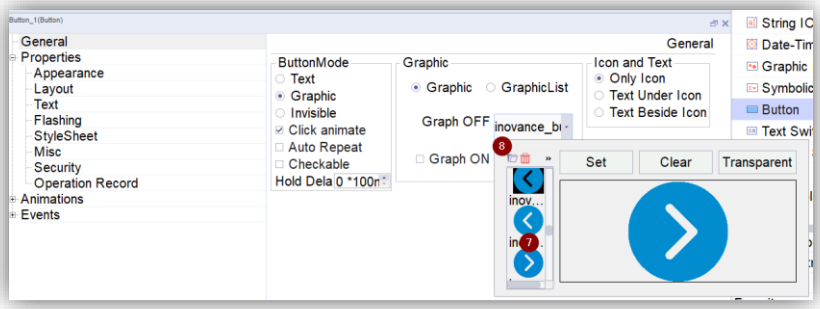
7.4.1 BUTTON

The "Button" component is used to insert a button with an image. In the properties of this component you can define the image of the button when it is not active and when it is active or select a list of images to change according to the value of a variable:

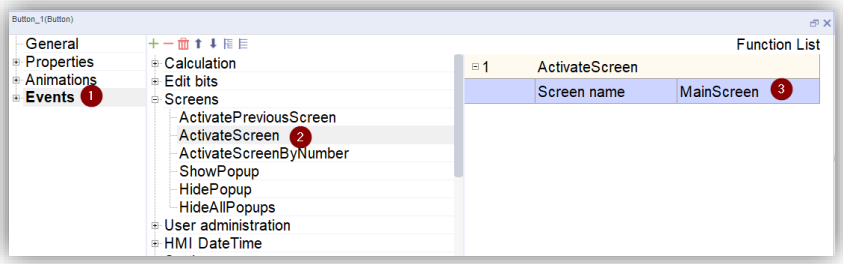


Follow the following process:

- Insert a " Button " component on the design screen (1).
- Select the "General" section of the component properties (2).
- Select “Graphic” (4) and “Graph” or “GraphicList” (5)
- To modify the button images, click on the drop-down (6).
- When opening the drop-down, you can select an image of the project from the dialog (7) or search for an image in another location by clicking on the folder button(8)

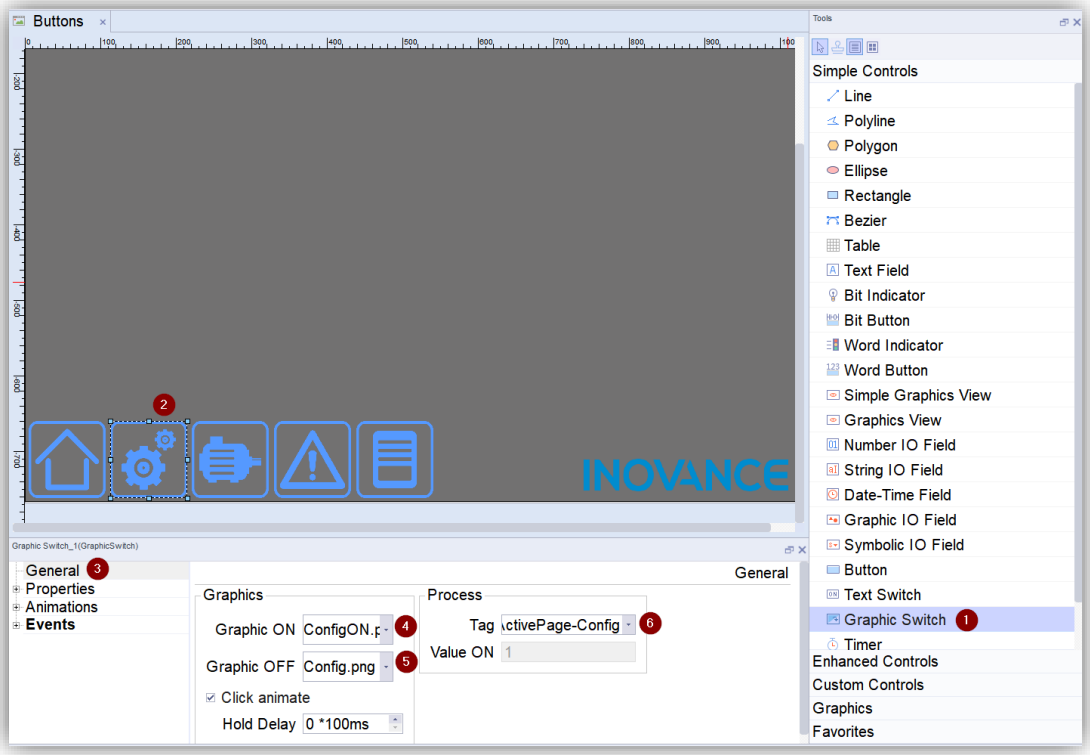


- The "Button" component cannot configure any variables that respond to click. To perform an action with this button you can configure the events.



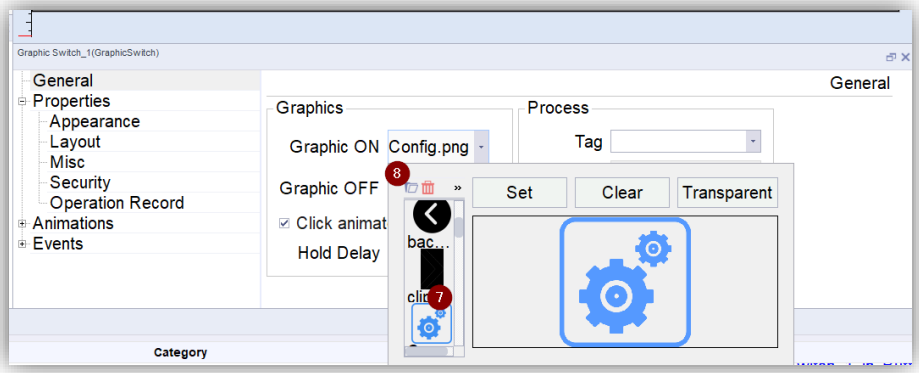
7.4.2 SIMPLE GRAPHICS VIEW

The "Simple Graphics View" component is used to insert a boon with an image. In the properties of this component you can define the image of the button when it is not active and when it is active:



Follow the following process:

- Insert a "Simple Graphics View" component on the design screen (1).
- Select the "General" section of the component properties (2).
- To modify the button images, click on the drop-down (4)(5).
- When opening the drop-down, you can select an image of the project from the dialog (7) or search for an image in another location by clicking on the folder button(8)



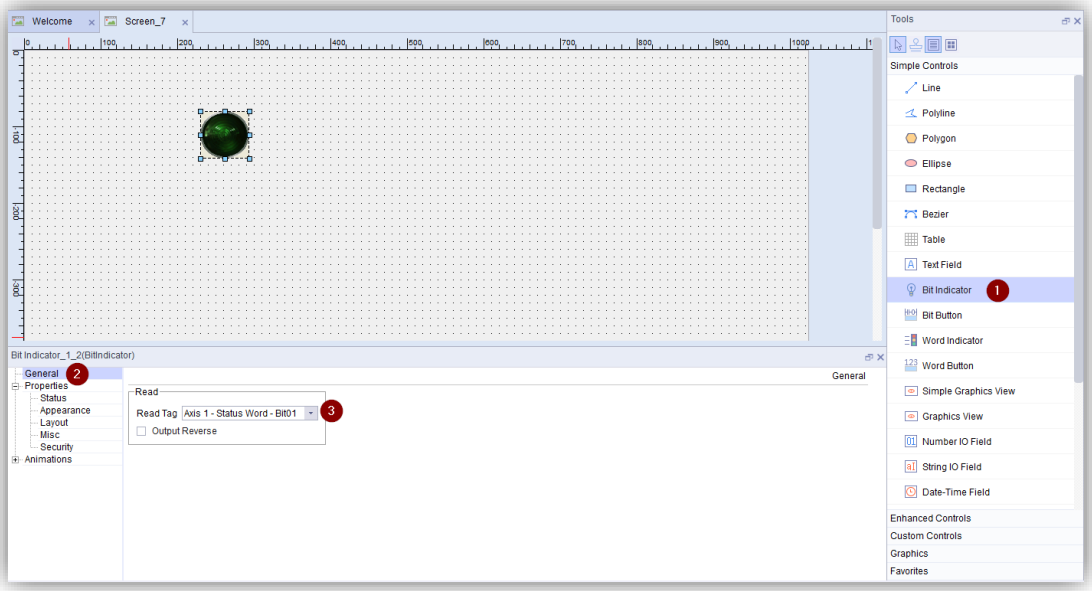
- Finally, a TAG (6) can be assigned to switch between the image assigned to the "Graphic ON" and "Graphic OFF" field. When the assigned TAG is 1, the image of the "Graphic ON" field is shown when the image of "Graphic OFF" is 0.

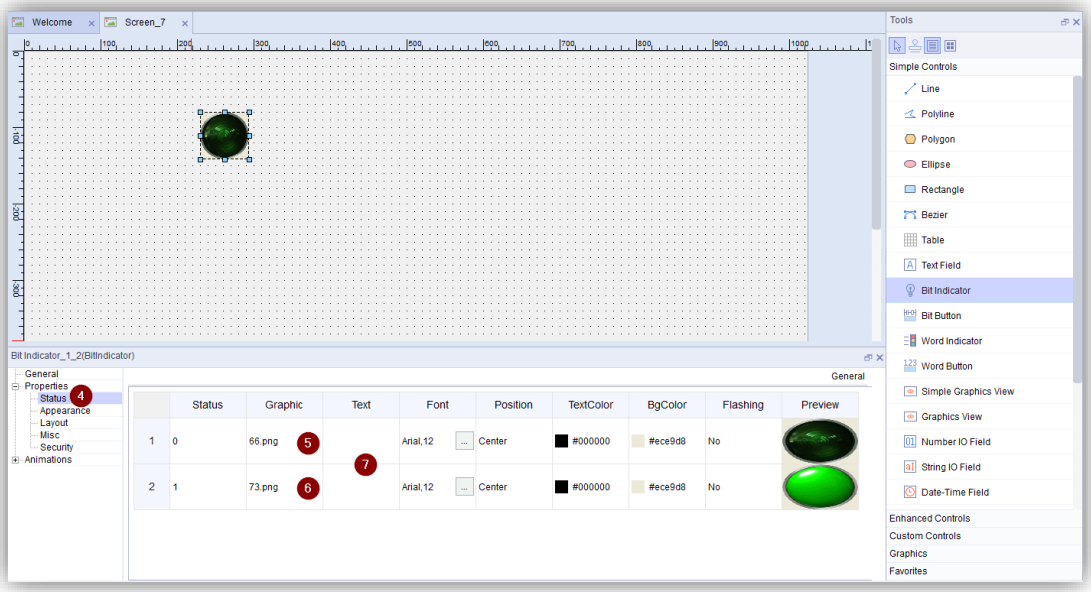
7.5 LAMPS

Lamps are widely used elements in HMI screen designs to show the status of a process bit. In this example we will show how to create a lamp in an InoTouchPro project.

Follow the following process:

- Insert a "Bit Indicator" component on the design screen (1).
- Select the "General" section of the component properties (2).
- Select the process variable of which we want to show the status (3).
- Select the "Status" section of the component properties (4).
- Select the graph for 0 state (5)
- Select the graph for 1 state (6)
- Remove state texts (7)

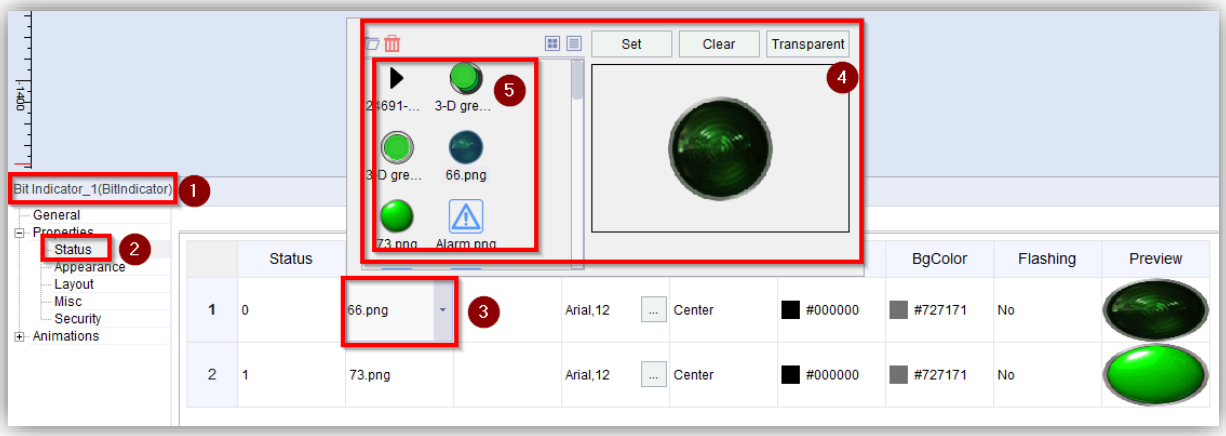




7.6 ADD GRAPHICS TO PROJECT

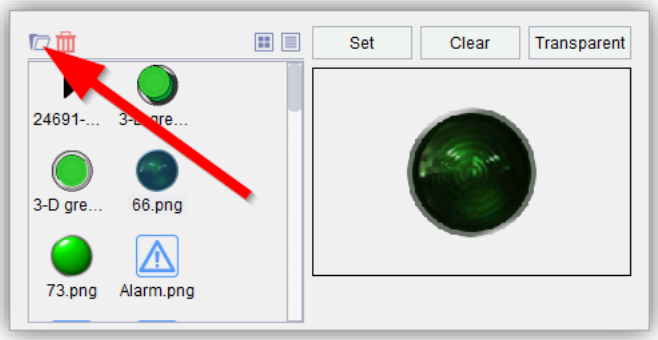
All components that use images or graphics have a property where you can choose the graphic that will display that component.

If you access this property of the component [1][2][3], in this case a “Bit indicator”, a small dialog box opens[4] that allows you to select an image from a predetermined list [5].

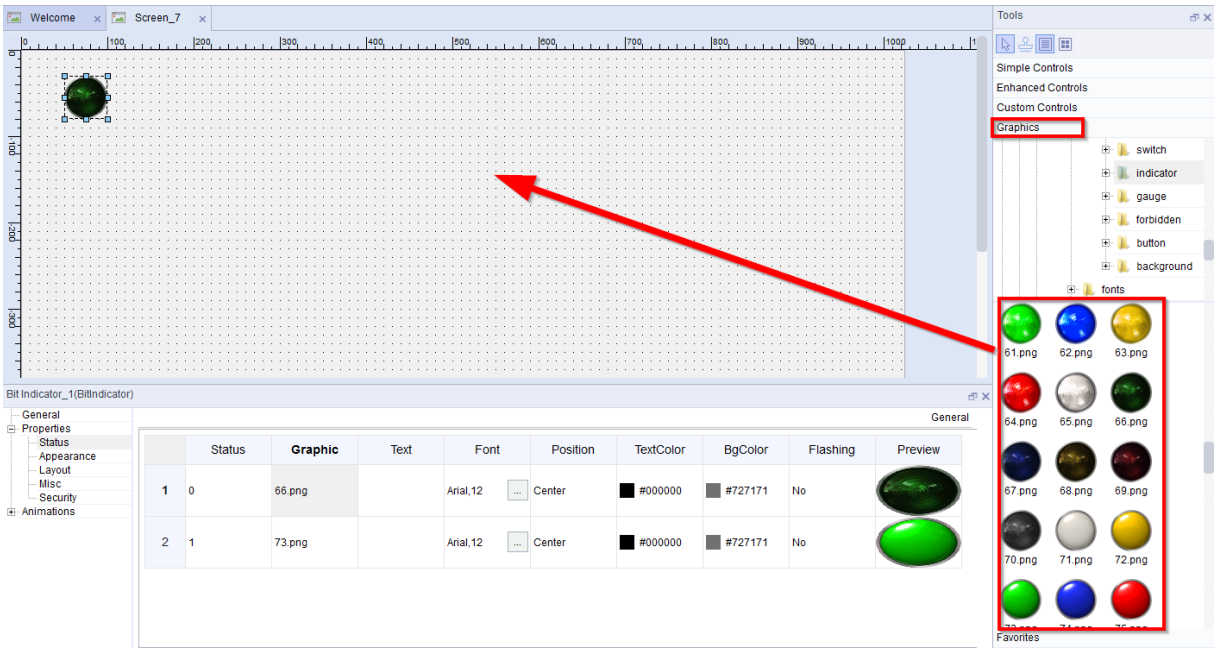


This list is made up of all the images that have been used in the current project. To add more elements to this list there are two possibilities:

- Use File Explorer to find an image on the computer's hard drive:



- Add an image from the InoTouchPad image library to the current project. After this the image will appear in the predefined list.


Welcome
Screen_7
Tools
Simple Controls
Enhanced Controls
Custom Controls
Graphics
switch
indicator
gauge
forbidden
button
background
fonts
61.png
62.png
63.png
64.png
65.png
66.png
67.png
68.png
69.png
70.png
71.png
72.png
Favorites
Bit Indicator_1(BitIndicator)
General
Properties
Status
Appearance
Layout
Misc
Security
Animations
Status
Graphic
Text
Font
Position
TextColor
BgColor
Flashing
Preview
1
0
66.png
Arial,12
Center
#000000
#727171
No
2
1
73.png
Arial,12
Center
#000000
#727171
No

8 PROJECT VARIABLES (TAGS)

The user uses InoTouchPad software to create an HMI project, which can create up to 64 variable groups, and the total number of variables in all variable groups does not exceed 32767 (excluding system variables)

8.1 EXTERNAL VARIABLES

Defines the variable that accesses the data register of the external device, called external variable, which is the image of some defined storage locations in the PLC.

These areas can be accessed by both HMI and PLC (that is, read and write operations). The data types supported by these areas depend entirely on the PLC device.

For example: Inovance inverter supports int16 and unint16 two data types, while Inovance H3U can support int16, unint16, int32, unint32.

8.2 INTERNAL VARIABLES

Variables that are not defined to access the data registers of external devices are called internal variables, which are images of some defined storage locations in the HMI.

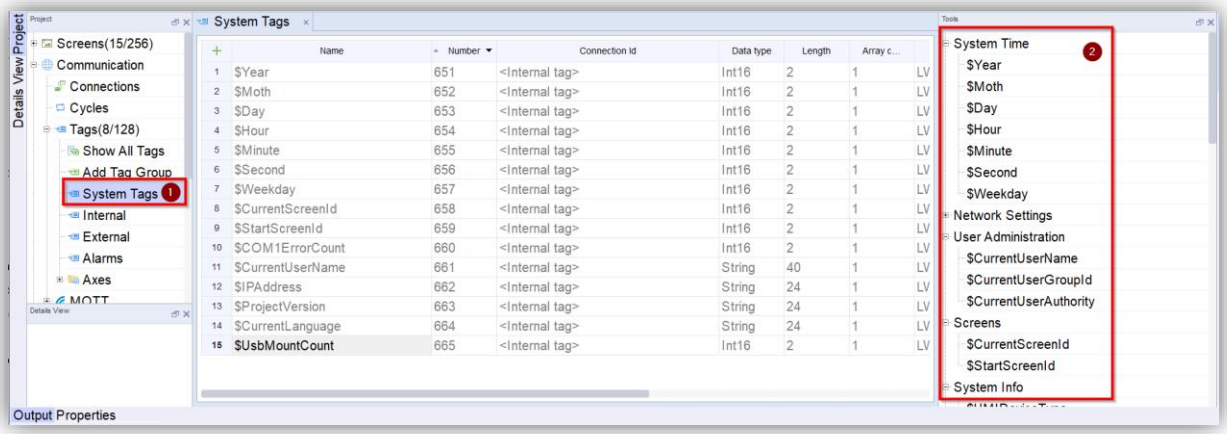
When HMI is used as a slave station, the storage area where internal variables are located can also be accessed by other devices.

8.3 SYSTEM VARIABLES

Internal variables are divided into some specific areas (LW 9000~LW 9323), which are used to display some system information of the system.

The addresses occupied by system variables are generally no longer used as ordinary internal variables, and the variables in this area are read-only variables.

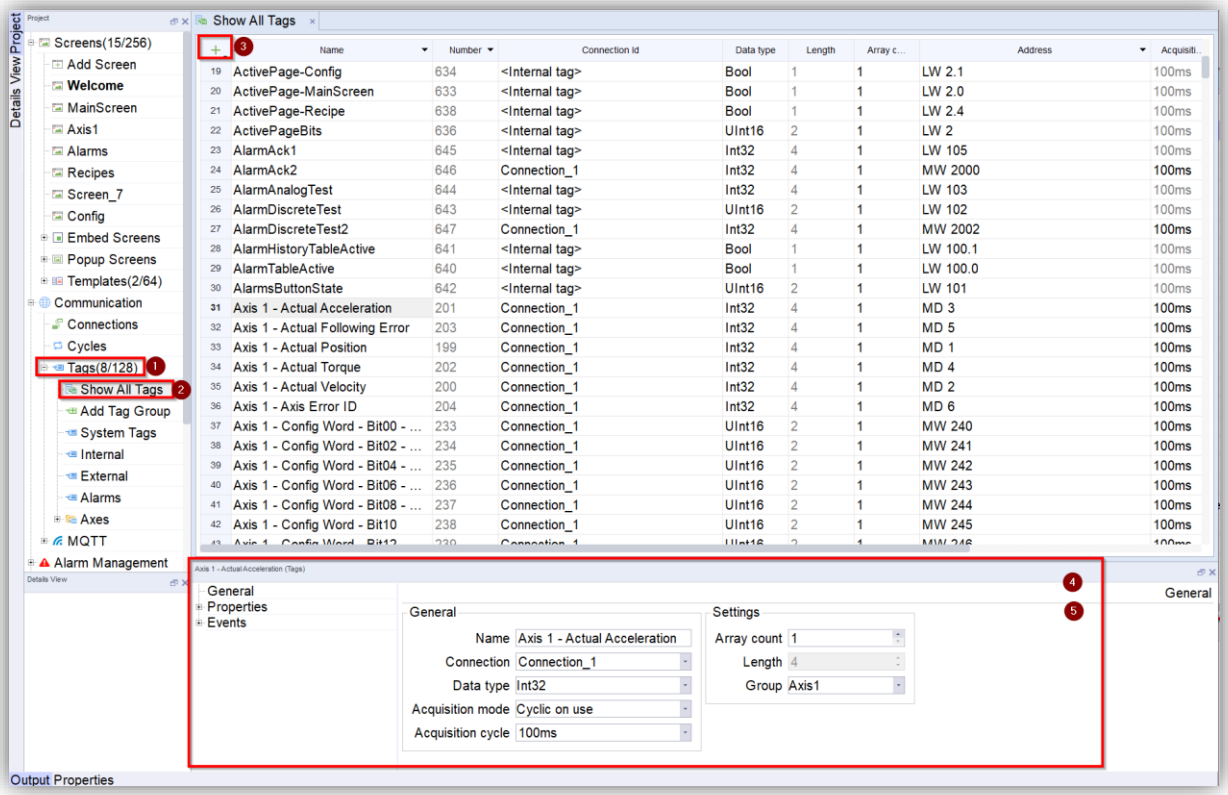
To use the system variables, they must first be added to the project by accessing the system variables screen(1) and adding the necessary ones from the variables tree on the right side of the screen(2):



8.4 CONFIGURATION VARIABLES

To add variables (TAGs) to the project, follow these steps:

1. Find the tree node "communication" in the project view and open the sub-options
2. Double-click to “Show All Tags” to open the variable editor.
3. Click the button in the workspace of the variable editor to create a new variable.
4. Then, according to user needs, select the connection, data type and address of the driver that needs to be connected to the PLC. If you want to use internal variables, you need to select the connection as the internal variable, and then select the data type and address.
5. Other attributes of variables can be set according to actual usage.



Limits

Variable limit values are only for numeric variables (except bool type). For string types (String, WString), date and time types (DateTime) the type limit value is invalid and cannot be set.

If the process value is higher or lower than the limit value range, this can be configured to trigger the alarm information or event. When the operator enters a value, if the value exceeds the limit, the value will be rejected and will not be saved.

Start value

The initial value of the variable is the value that will be preset when the runtime system starts. For all data types, an initial value can be preset. By setting the initial value, It can be ensured that when the project starts, the variables are in a user-defined state.

Logging

Configuration data recording refers to recording and storing the specified variable values, and its purpose is for subsequent data statistics, analysis and calculations.

Different record collection modes can be used:

- If the “on change” mode is used, the variable value will only be recorded once when the value of the variable changes once
- If the “on demand” is used, the variable value is recorded once after triggering a command (LogTag)
- If “cyclic continuous” is used, the value of the variable is recorded cyclically in the defined cycle time

Linear transformation

Through linear conversion processing, the value of the external variable can be scaled to adjust its value to the needs of the application

In order to perform a linear conversion of a variable, you first need to turn on the linear conversion switch in the variable editor, and then set the scaling parameters.

The upper and lower limits of HMI linear conversion and the upper and lower limits of PLC linear conversion are specified, and the numerical ranges will be linearly mapped to each other.

For example: create an external variable "variable _1", the selected connection for this external variable is the Inovance H3U monitoring protocol, and specify the access address as D0 address. Turn on the linear scaling, set the upper and lower limit values of the PLC to 10 and 0 respectively, and set the upper and lower limit values of the HMI to 100 and 0 respectively. When the HMI device is connected to the H3U , assuming that the value of "Variable_1" is written to 50 on the HMI device, it will be converted according to the linear relationship set before.

Then write the converted value into the D0 register in H3U (that is, the value written into the D 0 register in H3U is actually 5).

Similarly, in the HMI. Read the value of "Variable_1" from the device. If the value read from the D 0 register in H3U is 8, then after conversion according to the linear relationship, the actual display value of "quantity_1" on the HMI is 80.

For the linear conversion of variables, in addition to the above methods, the system functions LinearScaling() and InverseLinearScaling() can also be used.

Array variable

An array variable is a variable whose array count is greater than 1. It takes the specified address as the first address of the array, and the address space is continuous, and each array element has a phase.

The same data type, that is, the data type specified by the variable. Multiple array elements with the same properties can be addressed by a single array variable name, and then each array element can be used separately in the configuration.

In the InoTouchPad project, not all places can use array variables. Array variables can only be used under the following conditions:

- In the numerical IO field, array variable elements can be used.
- In the recipe editor, ingredient variables can be used.

9 COMMUNICATION

The data exchange between IT7000 and external devices is called communication. The external devices can be grouped through communication type such as serial communication and ethernet networks.

Network interconnection.

InoTouchPad maps the device register address to the communication variable by selecting the device protocol.

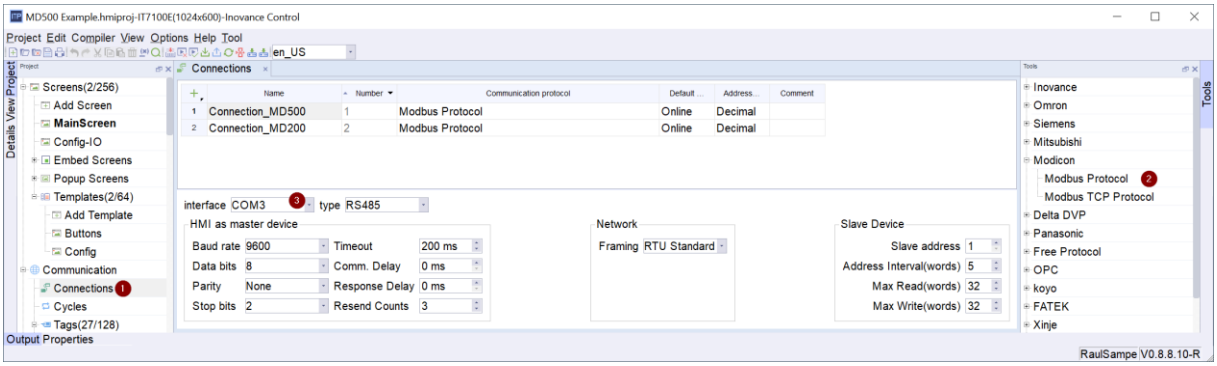
A communication device is any device that can be connected to a communication network and exchange data, usually called a communication node. HMI or PLC is usually considered as a communication node.

The data transmitted between devices can usually be used for the following purposes: for engineering control, collecting process data, reporting the status in the process, recording record process data.

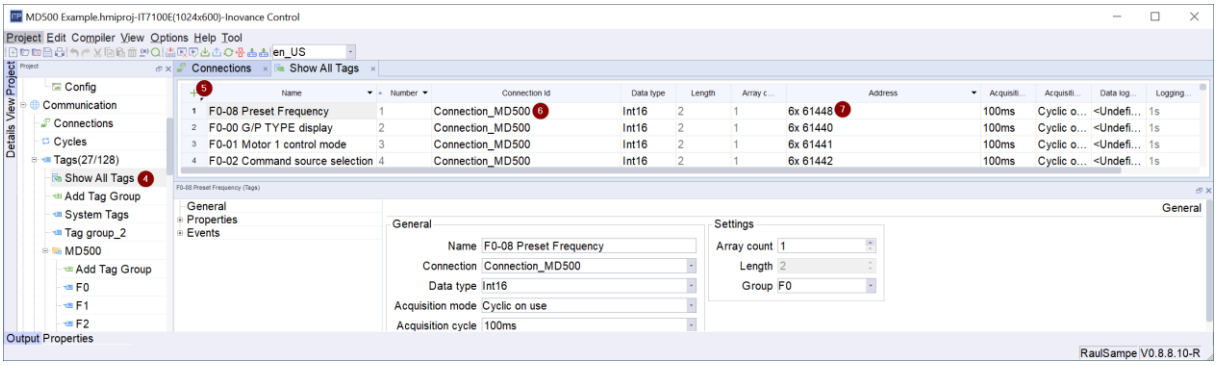
9.1 CONFIGURE COMMUNICATION

This is the procedure to setup a new communication:

1. Open the communication screen
2. Select the external device protocol
3. Setup the communications settings



4. Open TAGs configuration
5. Create a new TAG/Variable
6. Select the communication Id
7. Setup the slave variable address



Communication TAGs/variables are centrally managed in the "Tags" editor. They are divided into internal variables and external variables. The internal variables are defined internal variables on the HMI device. The external variable is a memory image defined on the PLC and can be used for communication. During the communication process, these defined memory images can be allow the device to access and implement read

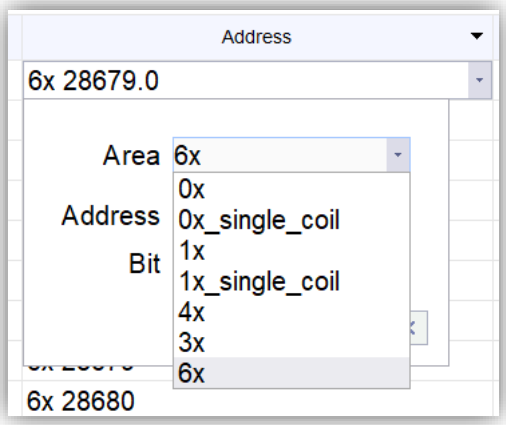
and write operations. The above operations are usually performed periodically or event-triggered. For details, please refer to 8 Project variables (TAGs).

9.2 COMMUNICATION PROTOCOLS

InoTouchPad has different drivers or protocols that allow it to connect with external devices. Users can choose different communication interfaces, configurations and transmission speeds for different devices.

9.2.1 MODBUS

The IT7000 implementation of the Modbus protocol uses the following function codes. Take into account that this codes are not the same as standard Modbus codes.



IT7000 Modbus protocol function codes:

Code	Description
4x	32bits read/write. This corresponds to reading the register using Modbus function code 0x03 and writing using 0x10.
5x	Same as 4x. The only difference is when reading double word, high word/low word will be reversed.
6x	16bits read/write. This corresponds to reading the register using Modbus function code 0x03 and writing using 0x06.
3x	16bits read only. This corresponds to reading the register using Modbus function code 0x04.
0x	1bit read/write. This corresponds to reading the coils status using Modbus function code 0x01 and force single coil 0x05. (Support multiple coils)
1x	1bit read only. This corresponds to reading the coils status using Modbus function code 0x01. (Support multiple coils)
0x_single_coil	1bit read/write. This corresponds to reading the coils status using Modbus function code 0x01 and force single coil 0x05.
1x_single_coil	1bit read/write. This corresponds to reading the coils status using Modbus function code 0x01.

Modbus Function code description:

Decimal	Hexadecimal	Description

01	0x01	Read Coil Status
02	0x02	Read Input Status
03	0x03	Read Holding Registers
04	0x04	Rear Internal Registers
05	0x05	Force Single Coil
06	0x06	Preset Single Register
15	0x0F	Force Multiple Coils
16	0x10	Preset Multiple Registers
22	0x16	Masked Write Registers

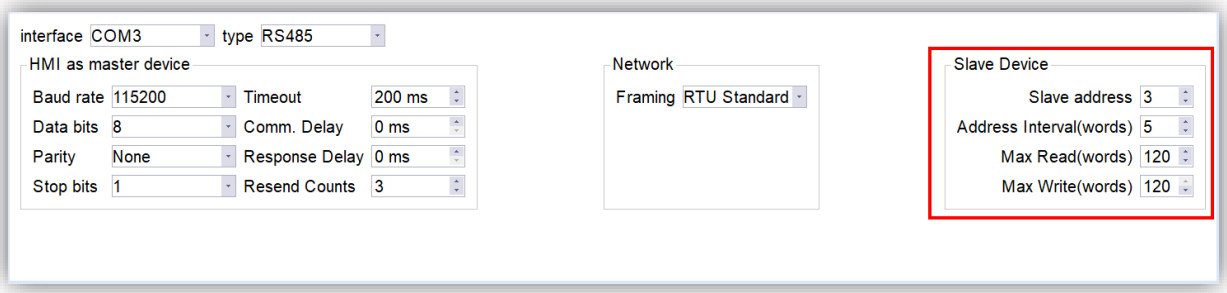
9.2.2 MD SERIES COMMUNICATION

The MD series drives use the Modbus RTU protocol to communicate with the IT7000 panel. Although it uses the standard Modbus RTU format it has some limitations. The protocol used in these drives can only send/receive 12 16-bit registers in the same frame. This limit must be taken into account when developing IT7000 projects.

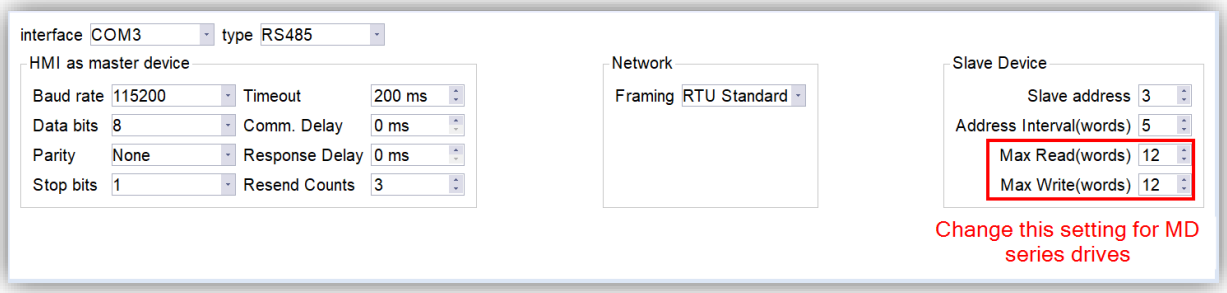
If in an IT7000 project there is a screen with more than 12 TAGs with a consecutive Modbus address, the screen will try to send these TAGs in the same frame. As the MD series drives have a limit of 12 TAGs an error will occur.

To avoid these communication errors, it is necessary to modify the communication settings of the HMI project. There are three parameters that allow us to modify the behavior of communications:

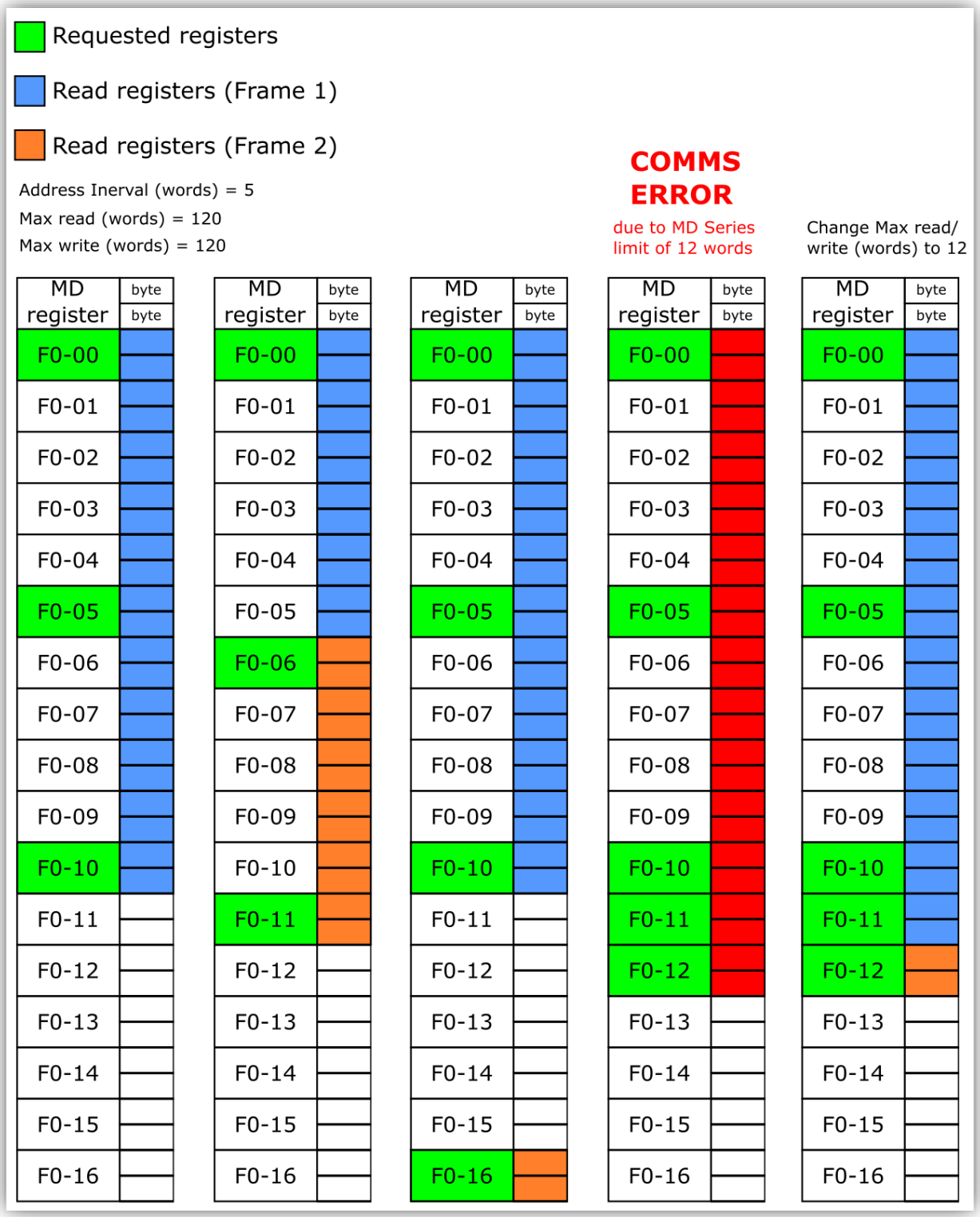
- **Address Interval (words)**: This parameter allows us to indicate when the TAGs are considered consecutive. In other words, if there are two TAGs whose Modbus address is separated by a maximum of the value of this parameter, those two TAGs are considered consecutive and will be sent in the same communications frame.
- **Max Read (words)**: Indicates how many words can be sent as a maximum in the same reading frame
- **Max Write (words)**: Indicates how many words can be sent as a maximum in the same write frame



NOTE Max read/write (words) parameters must be equal to 12 to eliminate communication errors with MD series drives.



The image below shows how communications behave with the MD series drives according to the configuration of these parameters:



9.2.3 PROTOCOL ERROR CODES

The error codes returned when the protocol communication is abnormal are as follows:

Protocol	Error code and description
Inovance Monitoring	0x10: Checksum error
	0x11: Data length error
	0x12: Unknown operation
	0x15: 15 error
Modbus family protocol	0x01: Illegal function
	0x02: Illegal data address
	0x03: Illegal data value
	0x04: Slave device failure
	0x05: Confirm
	0x07: The slave device is busy
	0x08: storage parity error
	0x0a: Unavailable gateway path
	0x0b: Gateway target device response failed

The system error codes returned when the system communication is abnormal are as follows:

	Error code and description
System communication error	0x10001: Device failed to open
	0x10002: Device write error
	0x10003: Read error
	0x10004: Read timeout
	0x10006: Bus is busy
	0x10007: Serial port identification error
	0x10008: connection offline
	0x10009: Write a command error
	0x1000a: Read command error once

10 ALARM MANAGEMENT

Visualization of process and system alarms

- Custom alarm: Configure alarms to display process status on the HMI device or measure and report process data received from the PLC.
- System alarm: System alarms are predefined on these devices to display specific system status in the HMI device.

10.1 CUSTOM ALARM

Available alarm process:

- Analog alarm: If a certain "variable" value exceeds the "limit" value, the HMI device will trigger an alarm. Up to 2000 analog alarms can be created.
- Discrete alarm: If the specified bit in a certain "variable" is set, the HMI device will trigger an alarm. And up to 2000 discrete alarms can be created.

Acknowledge the alarm:

All alarms may require an acknowledgment. Alarms can be acknowledged in two ways:

- confirm key, in the alarm view.
- through events

Discrete alarms can also be acknowledged by setting specific bits in from HMI variable

Alarm classes:

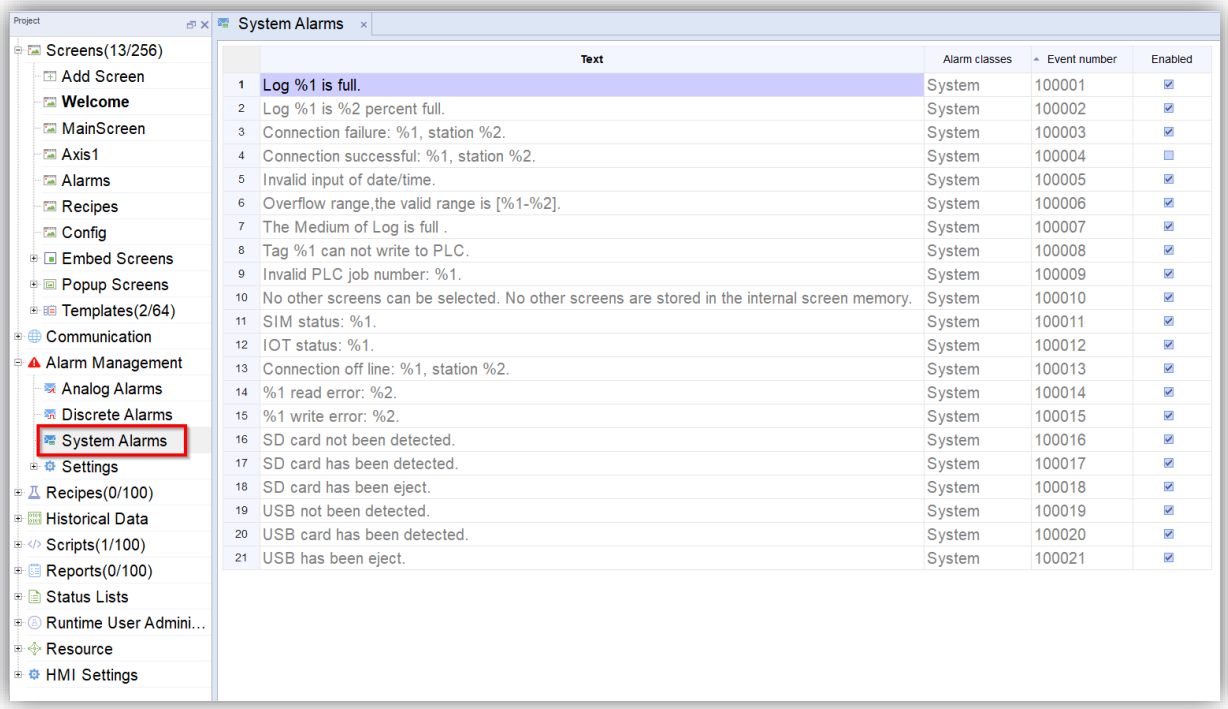
The alarm category mainly determines the display mode and acknowledgment behavior of the alarm on the HMI device. Up to 32 alarm categories can be created.

System predefined alarm category:

- "Error": Used for discrete and analog alarms to indicate emergency or dangerous operation and process status. This type of alarm **must be acknowledged**.
- "Warning" is used for discrete and analog alarms, indicating normal operation status, process status and process sequence. The alarms in this category **do not need to be acknowledged**.
- "System": Used for system alarms to remind the operator about the operating status of the HMI device and PLC. This alarm category cannot be used for custom alarms.

10.2 SYSTEM ALARM

The system alarm prompts the operating status of the HMI device and PLC, which is composed of a number and an alarm text. The reason for the alarm is precisely explained in the alarm text. The following are all predefined system alarms:

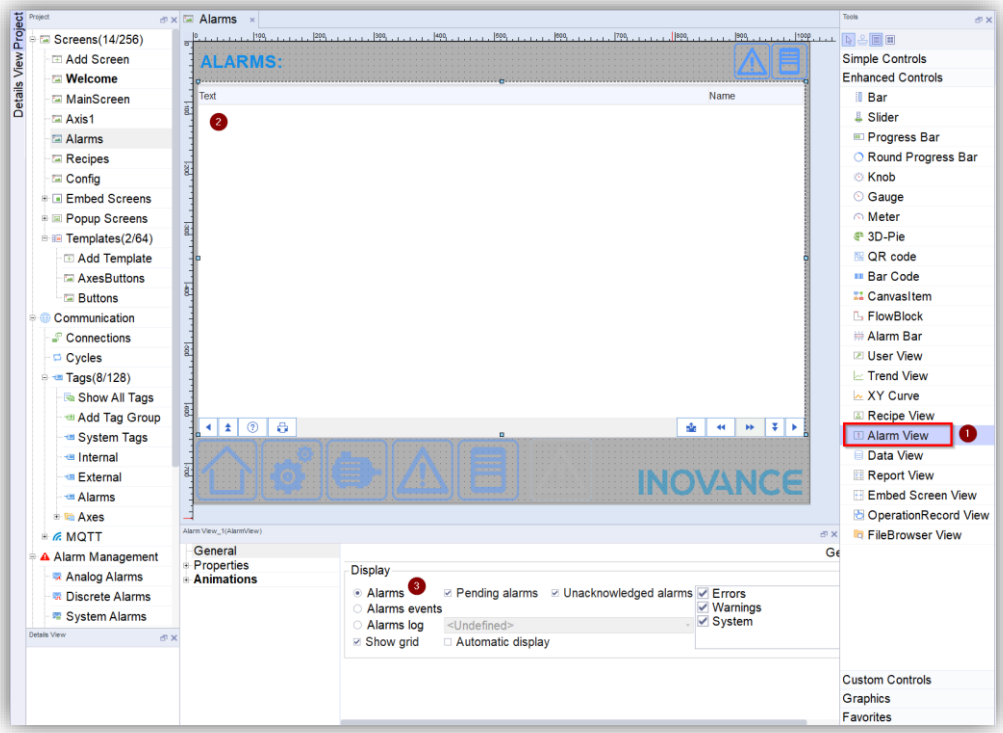


	Text	Alarm classes	Event number	Enabled
1	Log %1 is full.	System	100001	☑
2	Log %1 is %2 percent full.	System	100002	☑
3	Connection failure: %1, station %2.	System	100003	☑
4	Connection successful: %1, station %2.	System	100004	☐
5	Invalid input of date/time.	System	100005	☑
6	Overflow range,the valid range is [%1-%2].	System	100006	☑
7	The Medium of Log is full .	System	100007	☑
8	Tag %1 can not write to PLC.	System	100008	☑
9	Invalid PLC job number: %1.	System	100009	☑
10	No other screens can be selected. No other screens are stored in the internal screen memory.	System	100010	☑
11	SIM status: %1.	System	100011	☑
12	IOT status: %1.	System	100012	☑
13	Connection off line: %1, station %2.	System	100013	☑
14	%1 read error: %2.	System	100014	☑
15	%1 write error: %2.	System	100015	☑
16	SD card not been detected.	System	100016	☑
17	SD card has been detected.	System	100017	☑
18	SD card has been eject.	System	100018	☑
19	USB not been detected.	System	100019	☑
20	USB card has been detected.	System	100020	☑
21	USB has been eject.	System	100021	☑

10.3 SHOW ALARM

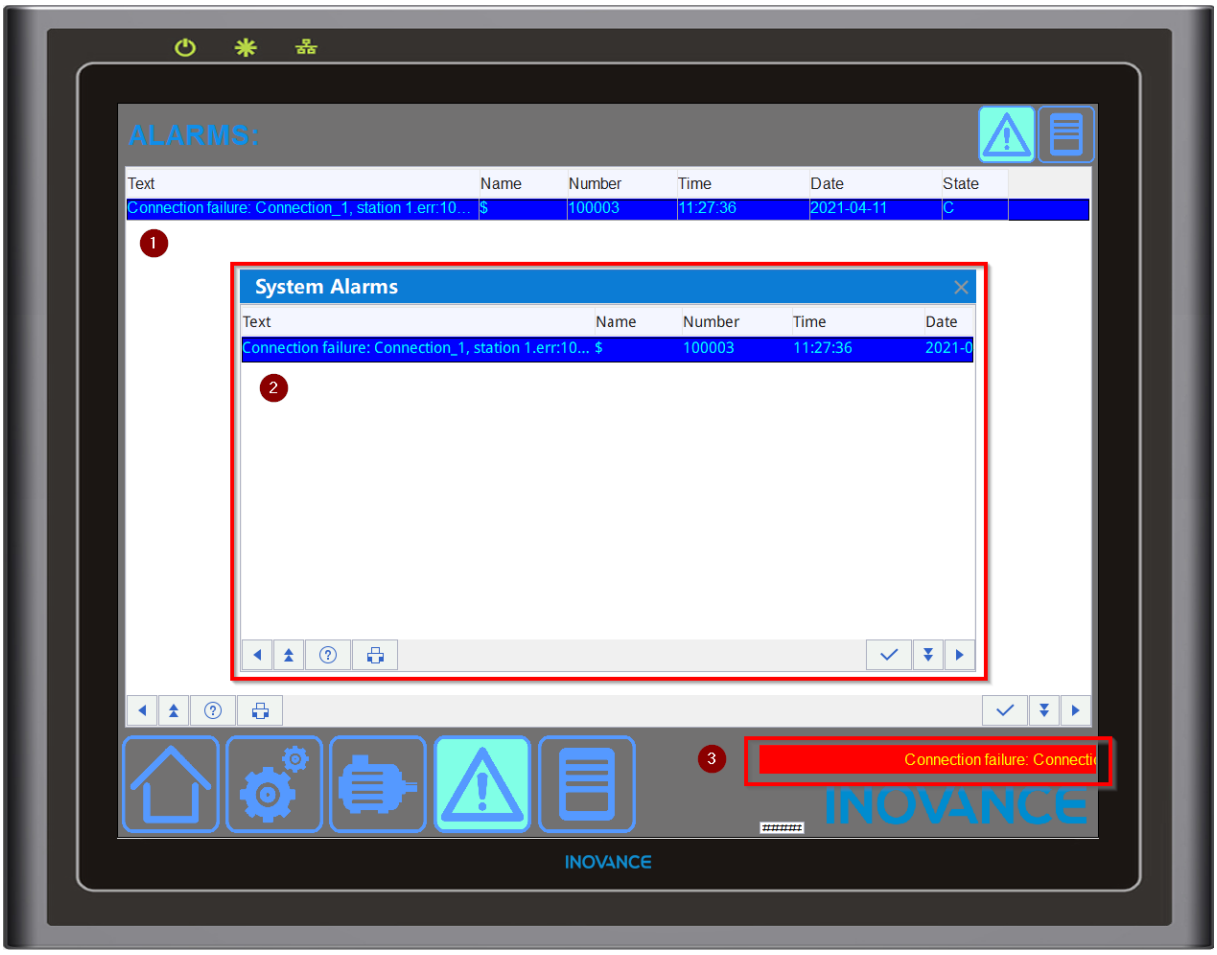
With the "Alarm View" and "Alarm Bar" components it is possible to display the status of the alarms and the alarm history. Inserting these components in a design screen allows to have a view of the alarms of the application.

The "Alarm View" component shows a table with the status and additional information of all alarms. From this component you can confirm the alarms and view additional information about them.



The following image shows the different alarm views:

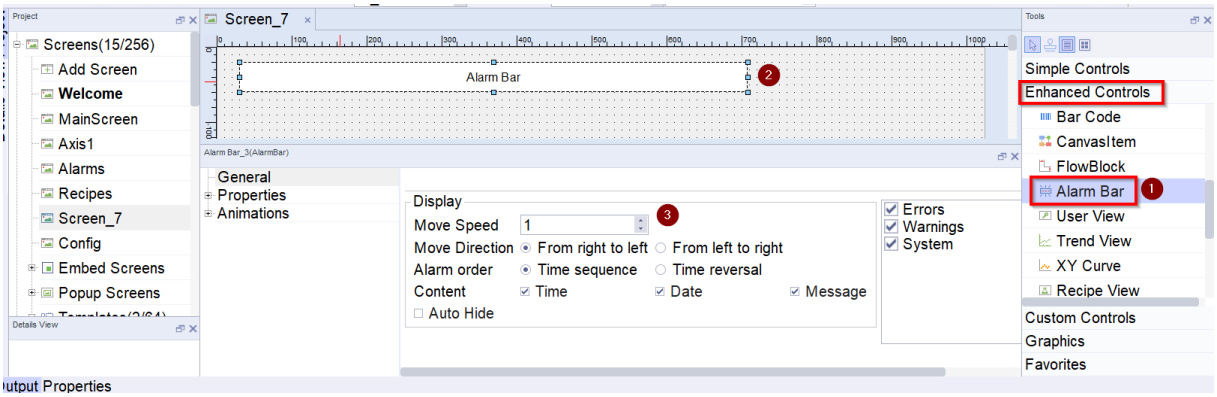
1. Alarm View
2. Pop-up View
3. Alarm Bar



10.3.1 ALARM BAR

The "Alarm Bar" component allows the alarms to be displayed in a single line of texts. If there is more than one active alarm, the text scrolls horizontally until it shows the information of all the configured alarms.

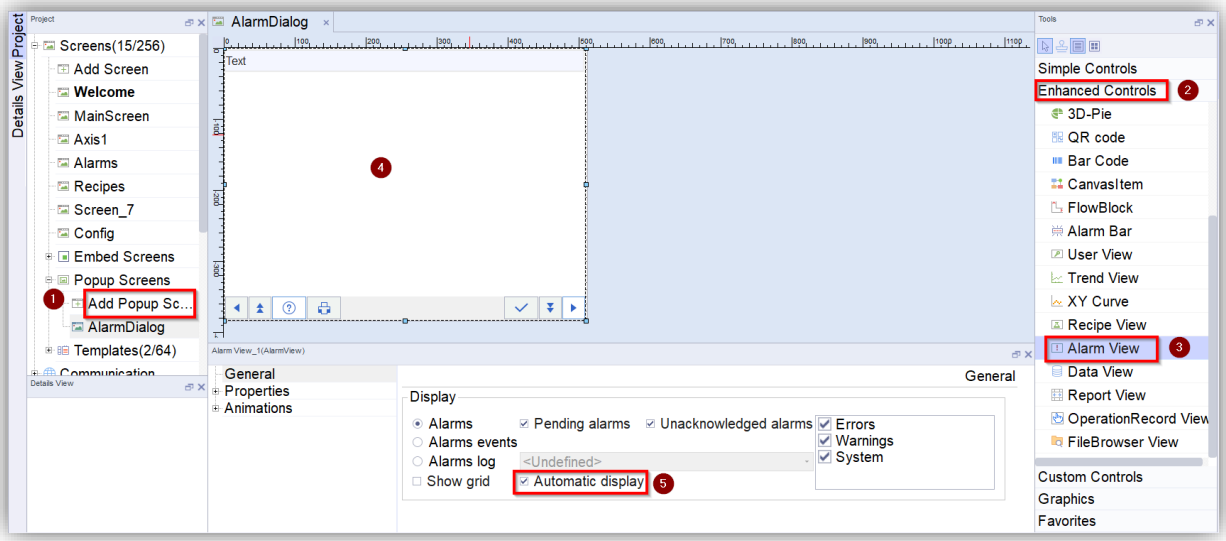
This component can be used in the header or base of a template to always have alarm information in view.



10.3.2 ALARM POP-UP

The pop-up alarm window allows you to display a pop-up window or dialog each time an alarm is triggered. This window is displayed above all other windows in the application.

The component for the pop-up window is the same as the normal alarm view "Alarm View", but this component must be added in a pop-up window instead of a normal window. To have the window appear automatically when an alarm is triggered, select the option

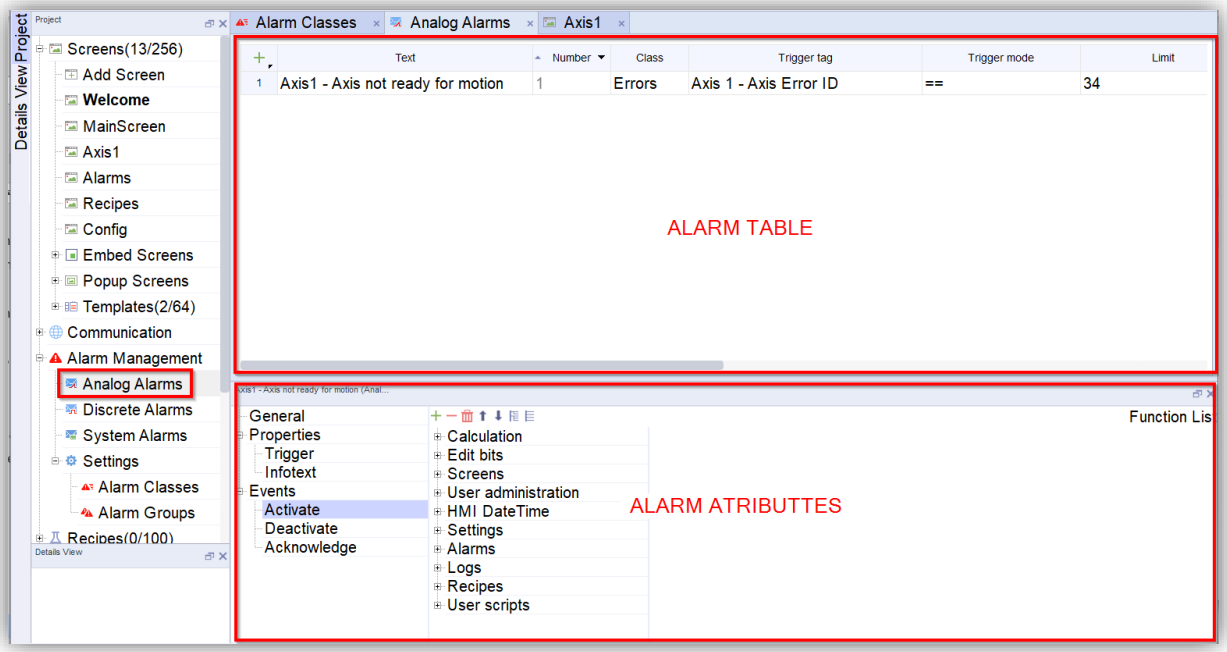


10.4 ALARM EDITOR

Using the alarm editor you can create analog alarms and specify their properties. To open the analog alarm editor interface click the "Alarm Management" item in the following project view, and then double-click "Analog Alarm" to open the editor.

"Analog Alarm" editor displays all the established analog alarms and their related attribute settings in table form.

Below alarm table is property view area. The attribute view provides the same attribute setting function as the table editing area. You can also define a series of events that occur when the alarm is triggered, deactivated or acknowledged.



Alarm attributes:

Alarm text: The alarm text is a description of the alarm, and the alarm text (analog + discrete) can be used as an international text.

Alarm number: The alarm number is used to identify the alarm. Each alarm number is unique among the following types. It is suitable for: analog alarm, Discrete alarm, system alarm.

Alarm class: The alarm class determines whether the alarm needs to be confirmed, the way the alarm is displayed on the HMI device, and the corresponding alarm log storage.

Trigger variables:

- analog alarm: the alarm is triggered when the value of the variable reaches the limit value
- discrete alarms: an alarm is triggered when a certain bit in the variable is set

Optional alarm attributes

Alarm group: If the alarm belongs to a certain group of alarms, you can acknowledge all alarms in this group by acknowledging only one alarm. Up to 16 alarm groups can be created.

Message text: The additional information corresponding to the alarm. The text will not be displayed automatically. After selecting the alarm, press the button  in the 'Alarm View Control' to display it.

Analog alarm

- Limit: compare the value of the trigger variable with the value to determine whether the alarm will be triggered.
- Trigger mode: there are >, <, ==, >=, <= five trigger modes, which are used as conditions for comparing the value of the trigger variable with the limit value.
- Hysteresis: hysteresis value, range 0~100, used in conjunction with percentage.
- Hysteresis percentage:

  When you select ON, the final hysteresis value is:

  hysteresis value = limit value x hysteresis %

  when you select OFF:

  hysteresis value = limit value

- Hysteresis mode
  - When "activated": value that triggers the alarm = limit value + hysteresis value
  - When "deactivated": value that triggers the alarm = limit value - hysteresis value

  The sign of the hysteresis value is determined by the trigger mode: when the trigger mode is >, >=, ==: the hysteresis value is positive; when the trigger mode is <, <=: the hysteresis value is negative.

- Delay: wait for the specified time before alarm is triggered. Time unit (ms).

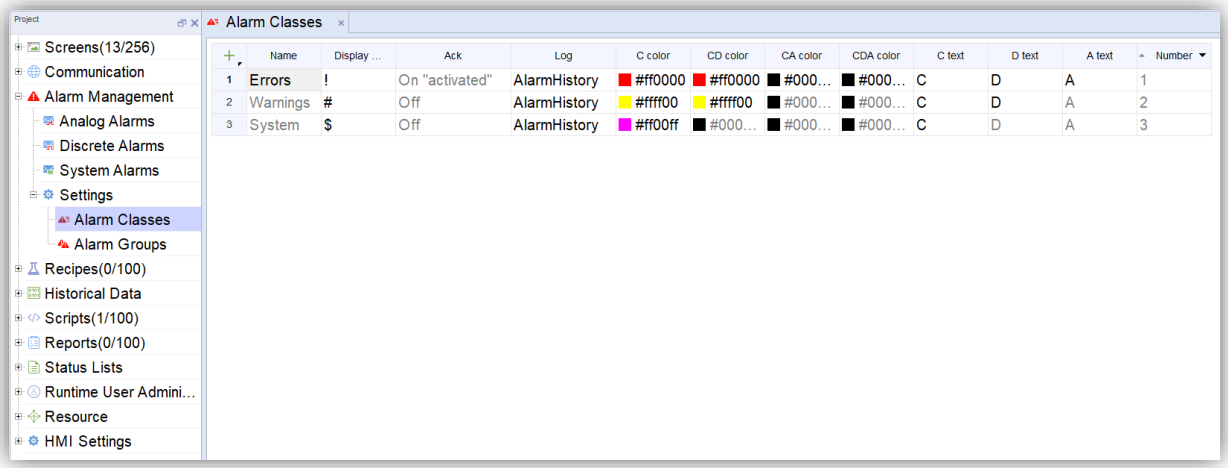
Discrete alarm

- Trigger bit: Indicates the bit of the trigger variable that will trigger the alarm.
- Trigger mode: There are four trigger modes, 1->0, 0->1, ==0, ==1, which are used as the condition of the trigger bit status change of the trigger variable
- Acknowledgment PLC variable: The PLC bit number specifies the nth bit of the PLC variable as the acknowledgment bit, and the corresponding alarm can be acknowledged by setting the specified bit of the PLC variable. PLC variables can only be the same as the trigger variable of the corresponding alarm, but the bit number should be different.
- Acknowledgment HMI variable: When the corresponding discrete alarm is confirmed, set the "HMI Ack bit" of the acknowledge HMI variable.
  NOTE Only when the discrete alarm triggers a PLC variable, the 'acknowledgment PLC variable' and 'Acknowledgment HMI variables' can be used.

10.5 ALARM CLASSES

In the "Alarm Classes" table editor, you can create alarm classes and specify their attributes.

Click "Alarm Management -> Settings -> Alarm Classes" in the following project view to open the "Alarm Classes" editor as shown in the following picture:



Property description:

Name: it is used to distinguish different alarm categories. The name in the alarm category is unique.

Display name: When an alarm is triggered, the label displayed under the 'Type' item of the "Alarm View Control". Different alarm categories can use the same display name.

Confirmation: Decide whether the related alarm needs to be confirmed.

Log: Determine the location of the alarms log or history.

Activated color: The color of the related alarm text displayed in the alarm view control when the alarm is only in the active state.

Activated and deactivated color: After an alarm is triggered and is eliminated, the color of the related alarm text displayed in the alarm view control.

Activated and acknowledge color: After an alarm is triggered and is acknowledged, the color of the related alarm text displayed in the alarm view control.

Activated, deactivated and confirmation color: the color of the relevant alarm text displayed in the alarm view after the alarm occurs, acknowledged, and eliminated.

C text (Activated): Status text after alarm is triggered.

D text (Deactivated): Status text after alarm is deactivated.

A text (Acknowledge): Status text after alarm is acknowledged.

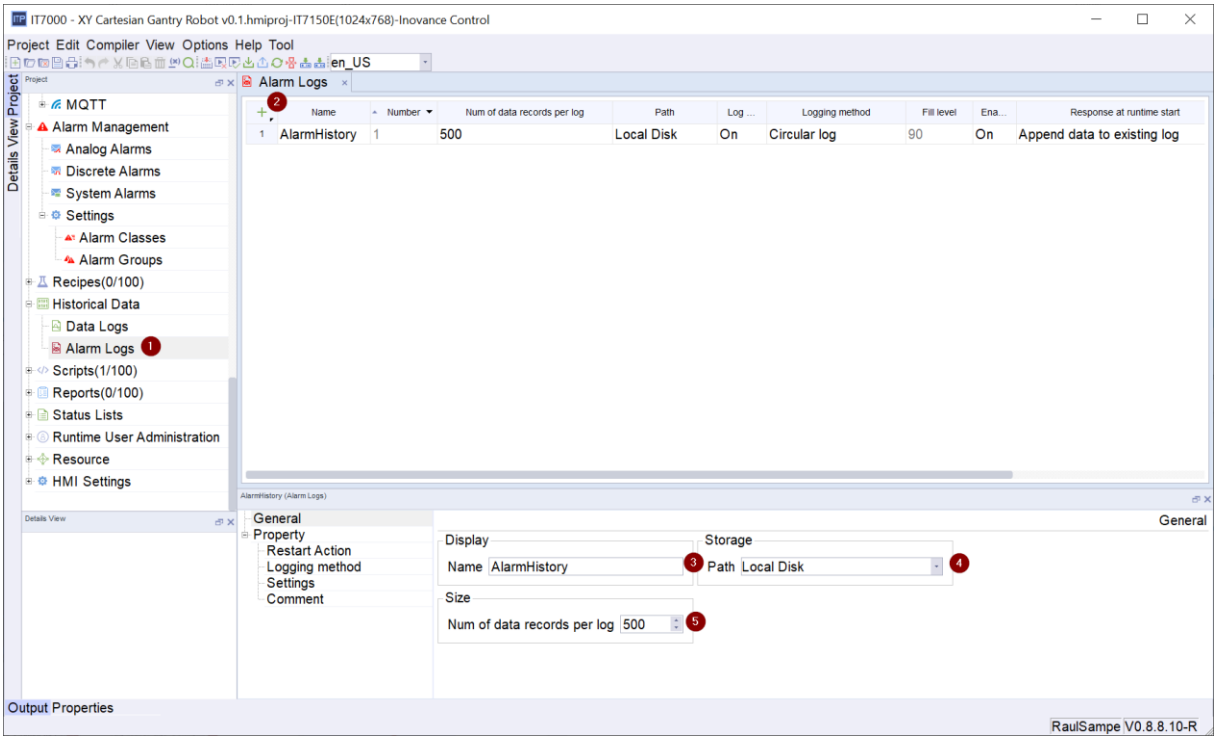
10.6 ALARM HISTORY

In order to display the alarm history it is necessary to create an "Alarms log". Different alarm logs can be created to keep history depending on the type of alarms.

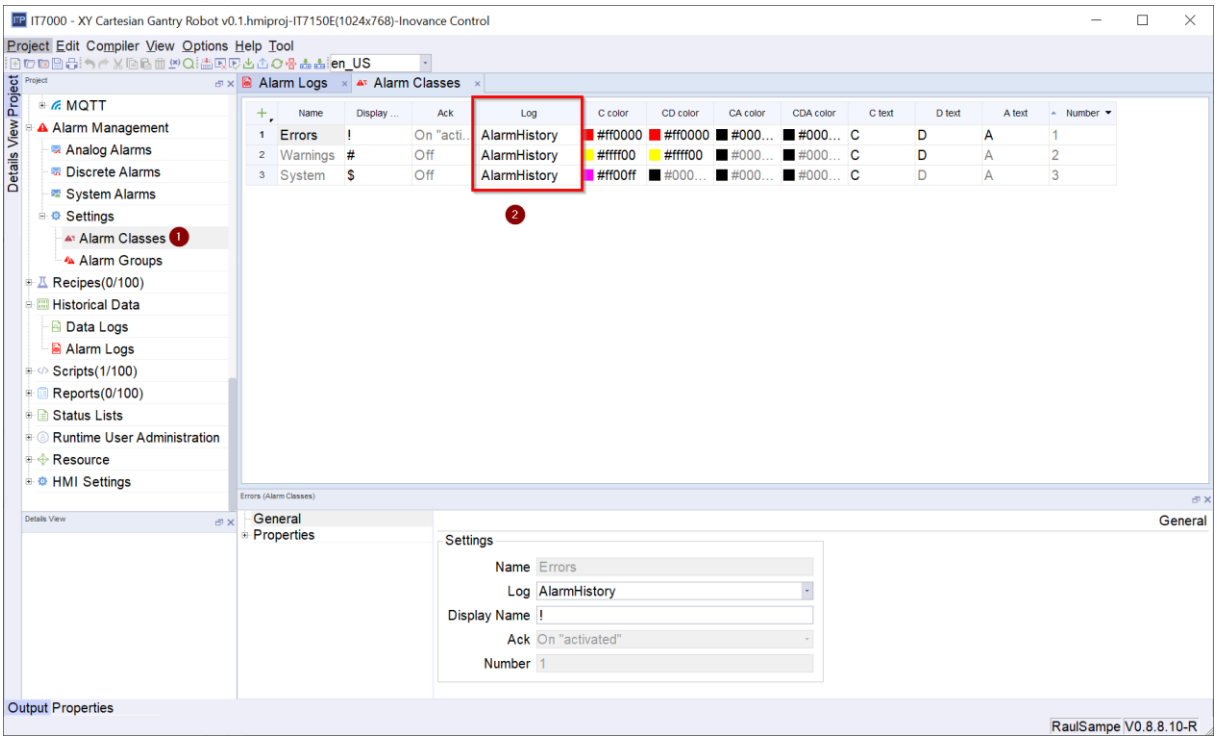
The type of alarms to be saved is defined in the alarm classes. Each alarm class can be stored in a different register.

Follow the procedure below to save the alarm log:

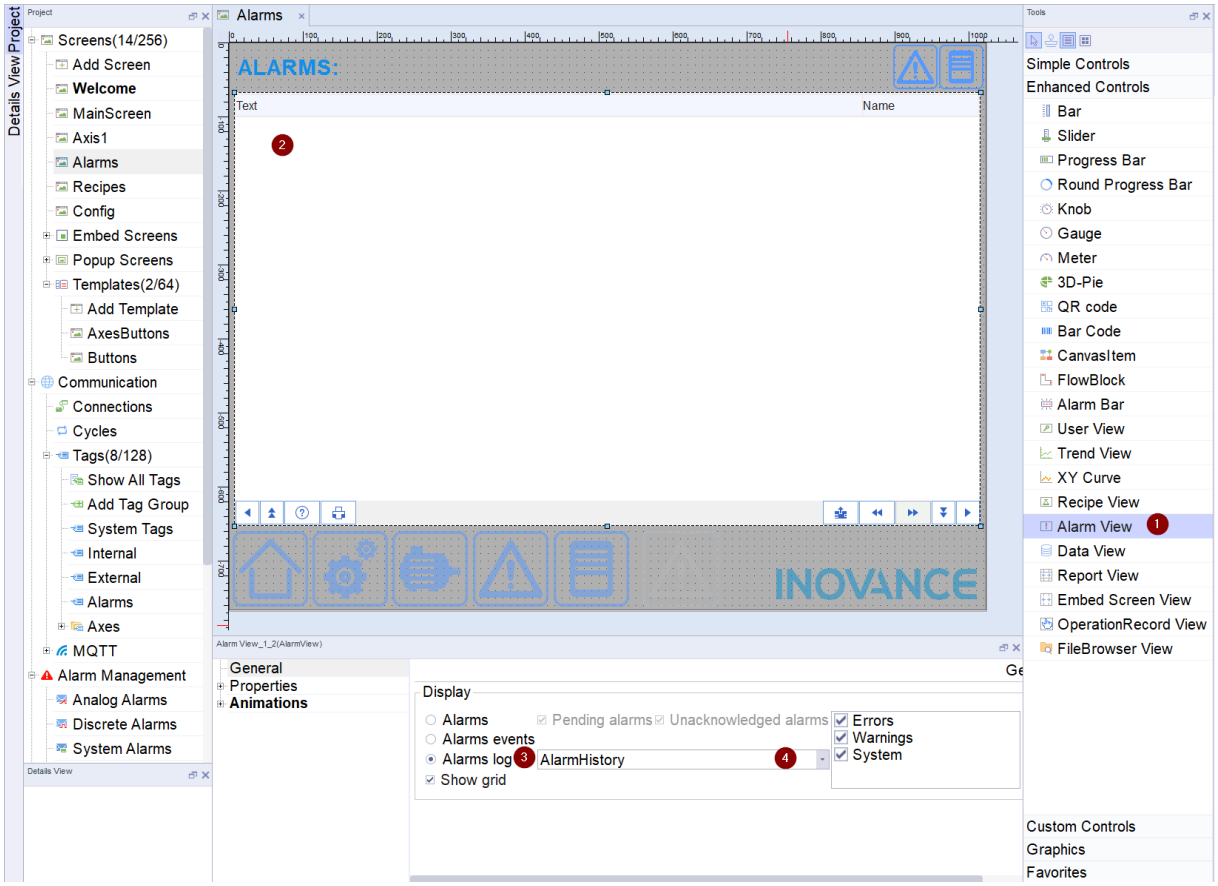
1. Create an "Alarms log" and define where the log is saved and the number of logs that will be saved



2. Assign to each type of alarm the record that has been created



3. Add the "Alarm View" component to a design screen, select the "Alarm log" option and finally select the "Alarm Log" where the alarms activity is stored.



11 RECIPES

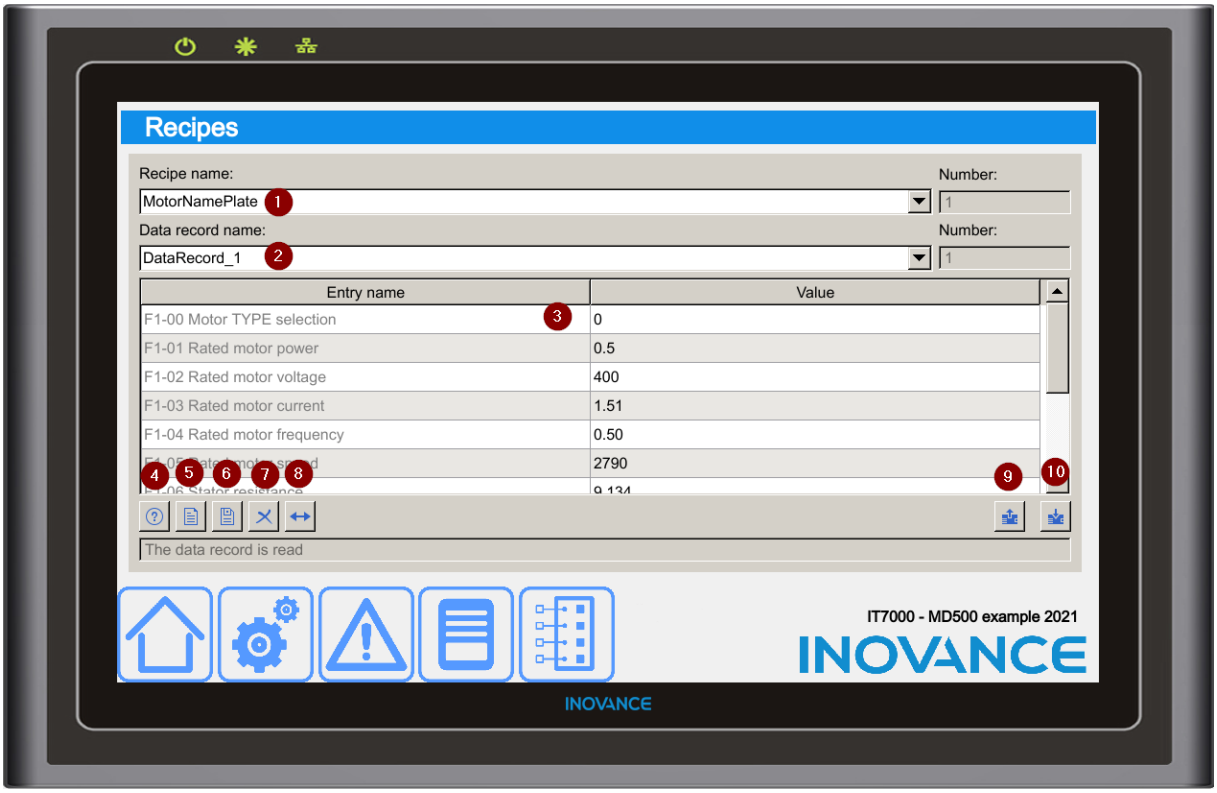
A recipe is a collection of the same type of data and has a fixed data structure. A recipe is made up of a set of process variables. From the recipe you can save / read a data record which is a file that saves the value of the variables that are part of the recipe.

The recipe can be stored on the HMI device or an external storage medium.

A project can configure up to 100 recipes, each recipe can configure up to 32767 ingredients, and each recipe can configure up to 1000 data records.

11.1 DISPLAY OF RECIPE

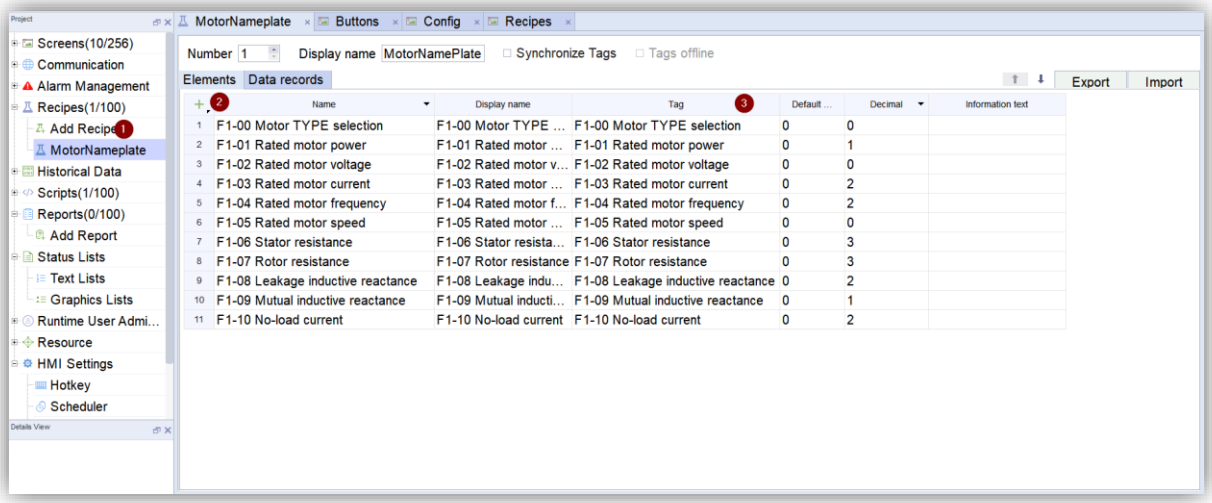
The recipe can be displayed through the recipe view control on the screen. The recipe view control is shown in the figure below:



1. Recipe selector.
2. Data record selector.
3. Recipe variables and values. These values can be edited.
4. It shows variable information text.
5. Create a new data record.
6. Save the variable values to the current data record.
7. Delete the current data record.
8. Synchronize the recipe variable values with the current data record. That is, copy the recipe variable values to the current data record.
9. Upload PLC variable values to the current data record.
10. Download data record values to PLC.

11.2 RECIPE EDITOR

The recipe editor allows you to configure the variables that are part of it. From the project tree you can create a new recipe and open the recipe editor:



Recipe configuration

Display name: name of the recipe displayed in the recipe view when the HMI device is running.

Synchronize Tags: When this option is enabled if the user selects a data record the values from this data record are copied to the recipe variables. If this option is disabled the user can modify the data record values without affecting the recipe variables.

If this option is enabled, and the user clicks on the button  in the recipe view, the values of the recipe variables are copied to the data record.

Tags Offline: If this option is enabled the recipe variable are not connected with PLC. To read/write the PLC variables the user has to click on the buttons  in the recipe view. If it is disabled, any change in the recipe variables is reflected in the PLC variables.

Recipe variables configuration

Name: unique identifier within the recipe.

Display name: name of the entry displayed in the recipe view when the HMI device is running.

Tag: HMI variables corresponding to recipe ingredients. Array variables can also be used here when the component variables are continuous addresses.

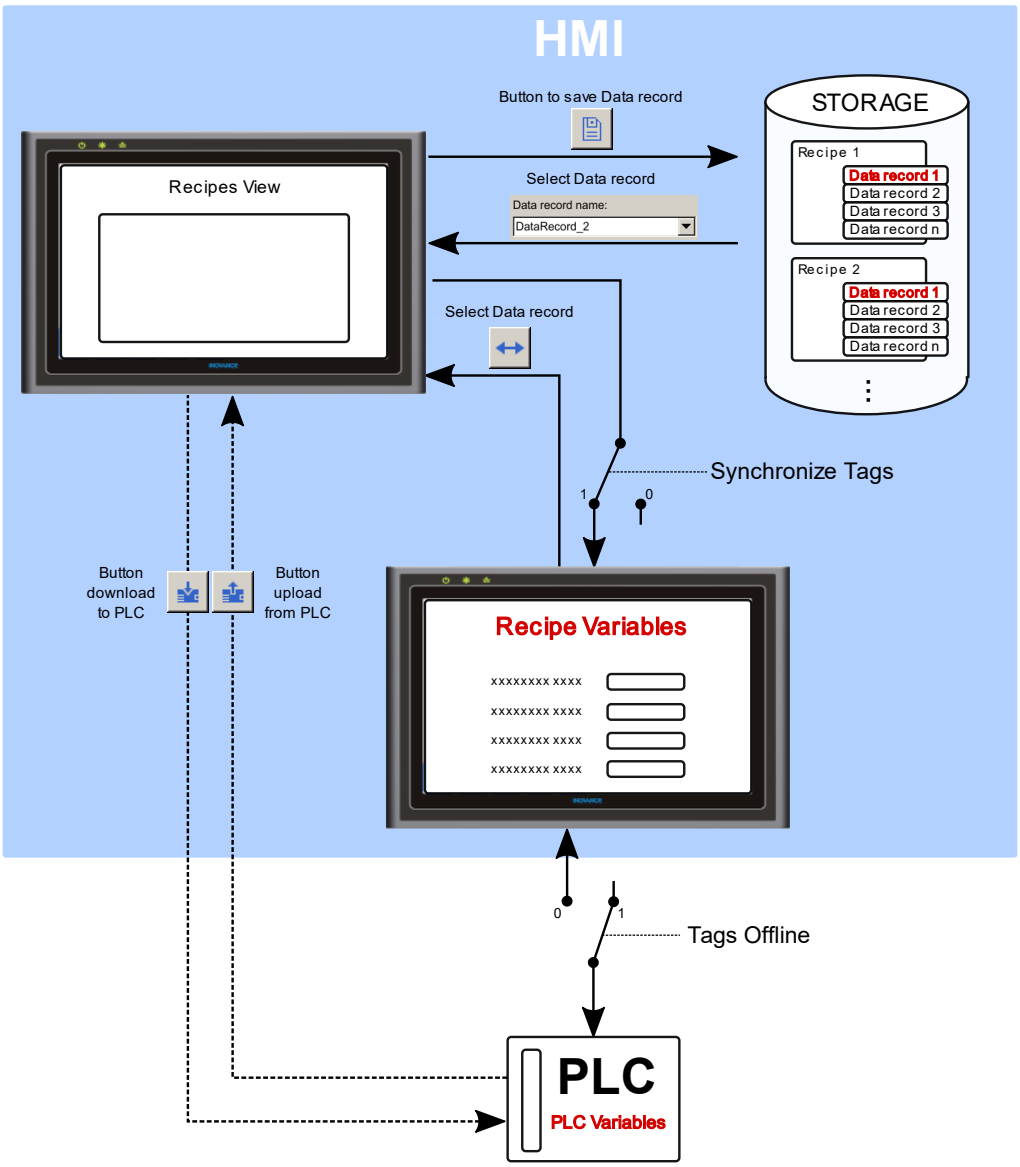
Default value: the default value of the recipe data.

Decimal point: Define the number of decimal places displayed in the recipe data during operation.

Information text: a description of the relevant recipe entry. When the HMI device is running, after selecting the corresponding entry, click on the recipe view info button

11.3 RECIPES BEHAVIOR

This section describes the behavior of recipe management on the IT7000 according to the configuration of each recipe applied in the InoTouchPad.



There are three distinct parts. The **data records**, the **recipe variables** and the **PLC variables**. The data records and the recipe variables are stored on the HMI. The PLC variables are the internal registers of the PLC to which the recipe variables are linked.

The three possible behaviors according to the recipe configuration are described below:

1. "Synchronize Tags" and "Tags Offline" are enabled. In this case, the user can modify the values of the data record and the recipe variables without affecting the PLC variables.
   Due to the option "Synchronize Tags" is active, when a data record is selected in the recipe view, the data from this data record is transferred to the recipe variables.

   If the user wants to copy the values of the recipe variables to the data record, he has to click on  in the recipe view.

   To read/write the PLC variables the user has to click on the buttons  in the recipe view

2. The second option is when 1 "Synchronize Tags" is enabled and "Tags Offline" is disabled. This mode behaves the same as the previous mode but any change in the recipe variables is reflected in the PLC variables.
3. The last mode is when both options are disabled. In this mode there is no connection between the data records and the recipe variables. The user can select the different recipe data records without modifying the recipe variables.
   To write/read the data record values to/from the recipe variables the user have to use the buttons  in the recipe view.
   In this mode, " Tags Offline " option is disabled, then any modification in the recipe variables is reflected in the PLC variables.

12 SIMPLE CONTROLS

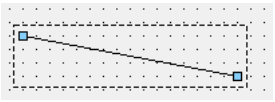
Below is a small summary of the simple controls that the InoTouchPad integrates.

12.1 LINE

Line

"Straight line" is an open object. The length and slope of the line are defined by the height and width of the enclosing rectangle.

The two end corners of the "straight line" each have a blue one, drag it to directly change its position.

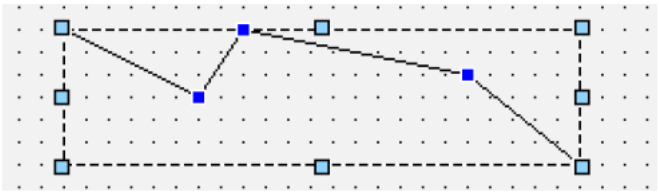


12.2 POLYLINE

Polyline

"Polyline" is composed of interconnected line segments and can have any number of corners (numbered one by one in the order of creation). "Polyline" every corner

Each has a blue dot, drag it to directly change its position. "Polyline" is an open object. Although the start point and end point may coincide at the same

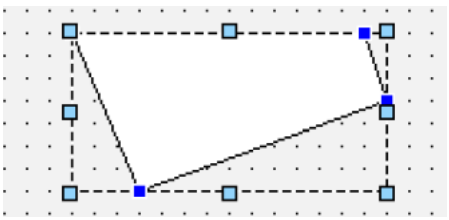


12.3 POLYGON

Polygon

A "polygon" is a closed object that can be filled with the background color. The corners of the "polygon" are numbered one by one in the order of creation. Each corner has its own

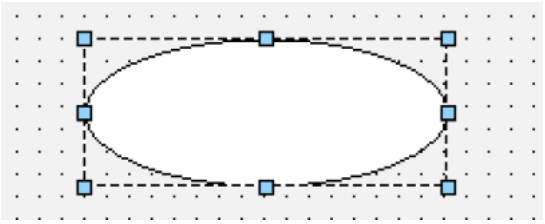
A blue point, drag it directly to change its position.



12.4 ELLIPSE

Ellipse

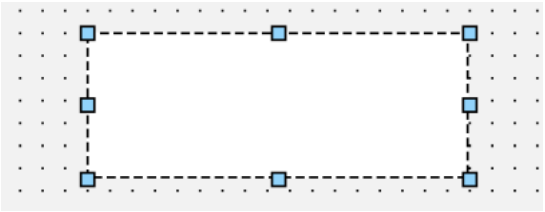
An "ellipse" is a closed object that can be filled with the background color.



12.5 RECTANGLE

Rectangle

A "rectangle" is a closed object that can be filled with the background color.

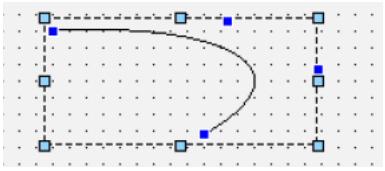


12.6 BÉZIER CURVE

Bezier

"Bézier curve" Displays the Bézier curve. The corners of the "Bézier curve" are numbered in the order of creation. Each corner has a blue dot,

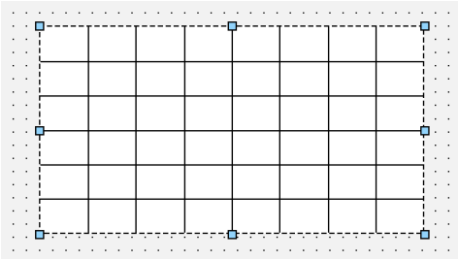
Drag it directly to change its position.



12.7 TABLE

Table

This component is used to display a grid. You can configure the number of columns and rows in this grid. It is a visual component that is not linked to any PLC or HMI variable.

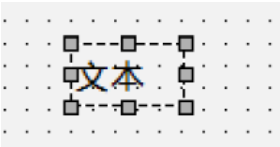


12.8 TEXT FIELD

Text Field

"Text field" is a closed object, you can enter one or more words in it, and you can define the font and color of the text, at the same time, you can also fill in. This object is like a label, is not connected to any tag.

Fill the background and border color of the text, you can enter text for all configured languages.

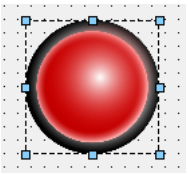


12.9 BIT INDICATOR

Bit Indicator

The "Bit indicator" is a component that shows two states depending on the value of a bit. The two states that this component shows can be of type text or graphic.

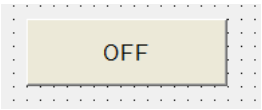
This component is very useful to use as a status led. A led can be shown on or off depending on the state of the associated variable.



12.10 BIT BUTTON

Bit Button

The "Bit button" is a button that changes the value of a bit of the PLC or HMI. Depending on the value of this bit, the button can change its text or the graphic shown.

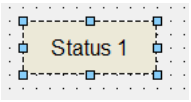


12.11 WORD INDICATOR

Word Indicator

The "Word indicator" is a component that shows different states depending on the value of a word. The states that this component shows can be of type text or graphic.

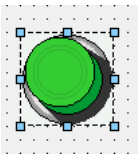
This component is very useful to show the status of any element of the PLC or HMI in text or graphic format. Unlike the "Bit Indicator" this component can show more than two states.



12.12 WORD BUTTON

Word Button

The "Word button" is a button that changes the value of a word of the PLC or HMI. Depending on the value of this word, the button can change its text or the graphic shown.

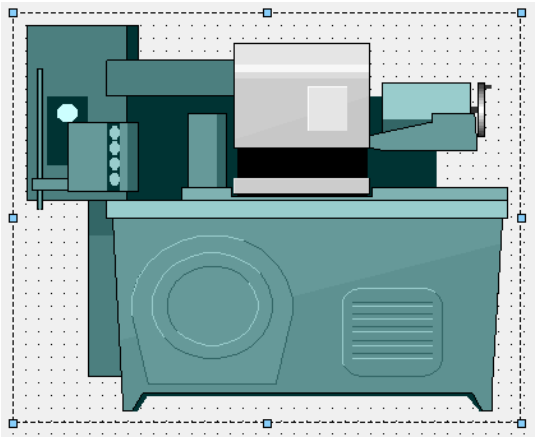


12.13 SIMPLE GRAPHIC VIEW

Simple Graphics View

The "graphic view" is used to display graphics. This component is for displaying static graphics. This can be hidden / shown and moved.

This component is not directly associated with a process variable.

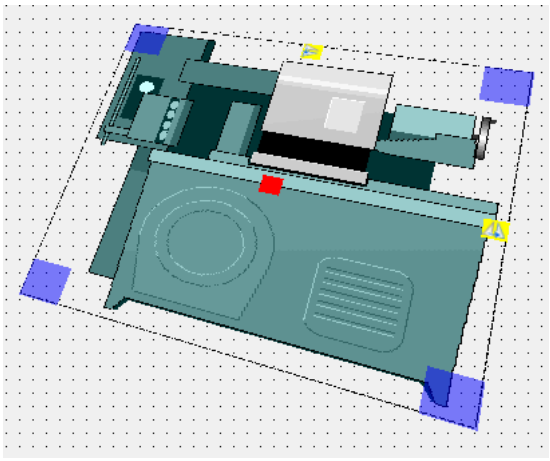


12.14 GRAPHIC VIEW

Graphics View

The "graphic view" is used to display graphics. This component is for displaying static graphics. This can be hidden / shown and moved. Unlike the "Simple Graphics View" component, it can modify the appearance of the displayed image.

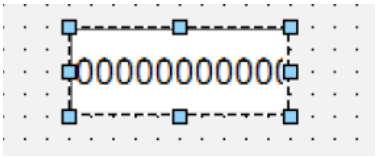
This component is not directly associated with a process variable.



12.15 NUMERICAL IO FIELD

Number IO Field

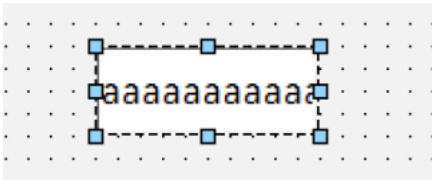
"Numerical IO field" is used to enter and display numerical process values.



12.16 STRING IO FIELD

String IO Field

"String IO field" is used to enter and display string process values.



12.17 DATE-TIME FIELD

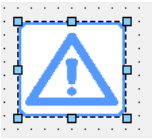
Date-Time Field

"Date-time field" is used to input and display the system time and date.

12.18 GRAPHIC IO FIELD

Graphic IO Field

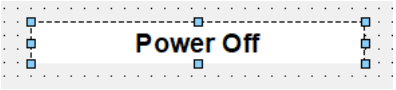
This component allows displaying a series of images according to the value of the associated process variable. Each image is associated with a value of the process variable. The images used in this component have to be in an image list previously created in the current project.



12.19 SYMBOLIC IO FIELD

Symbolic IO Field

This component allows displaying a series of texts or messages according to the value of the associated process variable. Each text is associated with a value of the process variable. The texts used in this component have to be in an text list previously created in the current project.



12.20 BUTTON

Button

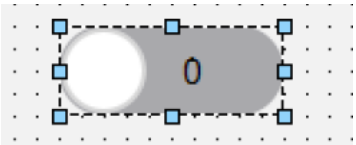
The "Button" is used to generate an event in response to the user's click. For example, it can be used to switch from one screen to another.

It can display a text or a graphic, and it can be invisible to create a touch zone on the screen.

12.21 TEXT SWITCH

Text Switch

"Text switch" is used to switch between two predefined states during operation. It can display a different text or graphic depending on the state of the "switch"

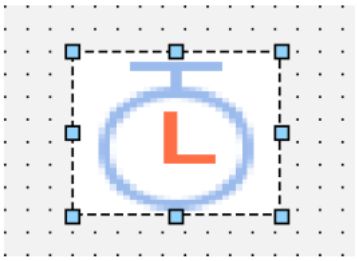


12.22 TIMER

Timer

"Timer" is used to configure the timing trigger event, which can be triggered on a single or cyclical basis. The configured timer only has an effect on the configuration screen where it has been added.

When the timer screen is displayed, the event is triggered. When you exit the timer screen, it no longer takes effect. The time can be configured in the properties of the component.



13 USER MANAGEMENT

13.1 FUNCTION OVERVIEW

When the system is being used, to prevent malfunction or manipulation of essential parameters, it is often necessary to introduce user management (that is, to manage user authority) to achieve differentiated access.

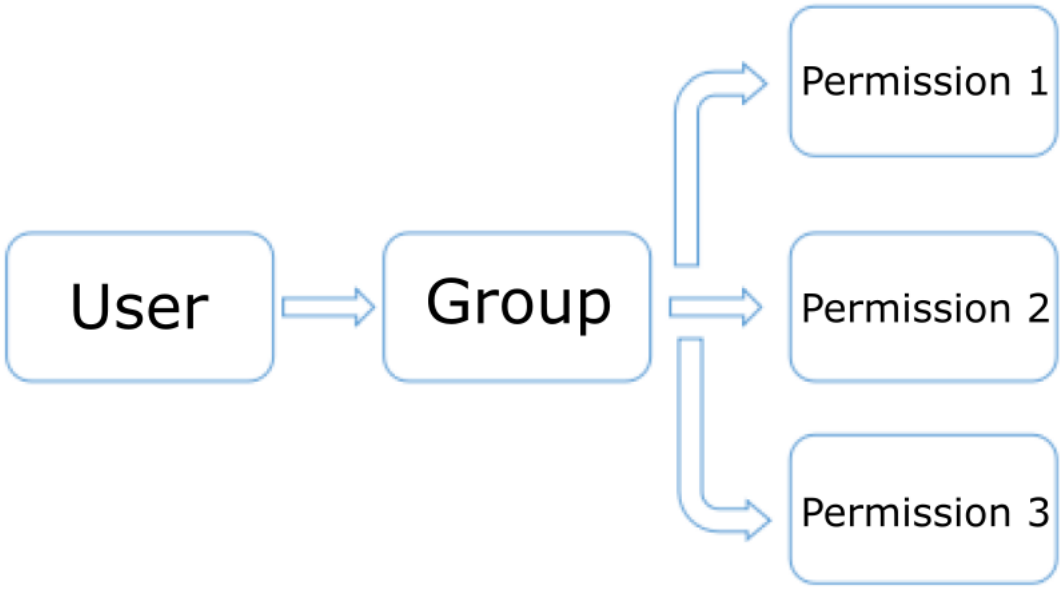
In the InoTouchPad configuration software, user management focuses on the access protection of data and functions.

After creating users and user groups and assigning permissions, the user can configure permissions for the objects in the screen editor. After they are transferred to the HMI device, all permissions are configured

The screen object should be protected, and the user can only access it if the login is successful and authorized.

13.2 BASIC PRINCIPLES

In user management, permissions are not directly assigned to users, they are assigned to user groups. For example, a user named "User 1" is assigned to "Operator Group" and he is granted corresponding permissions. It is not necessary to assign permissions to each user separately, assigning permissions to the user group is enough.



13.3 ELEMENTS AND BASIC SETTINGS

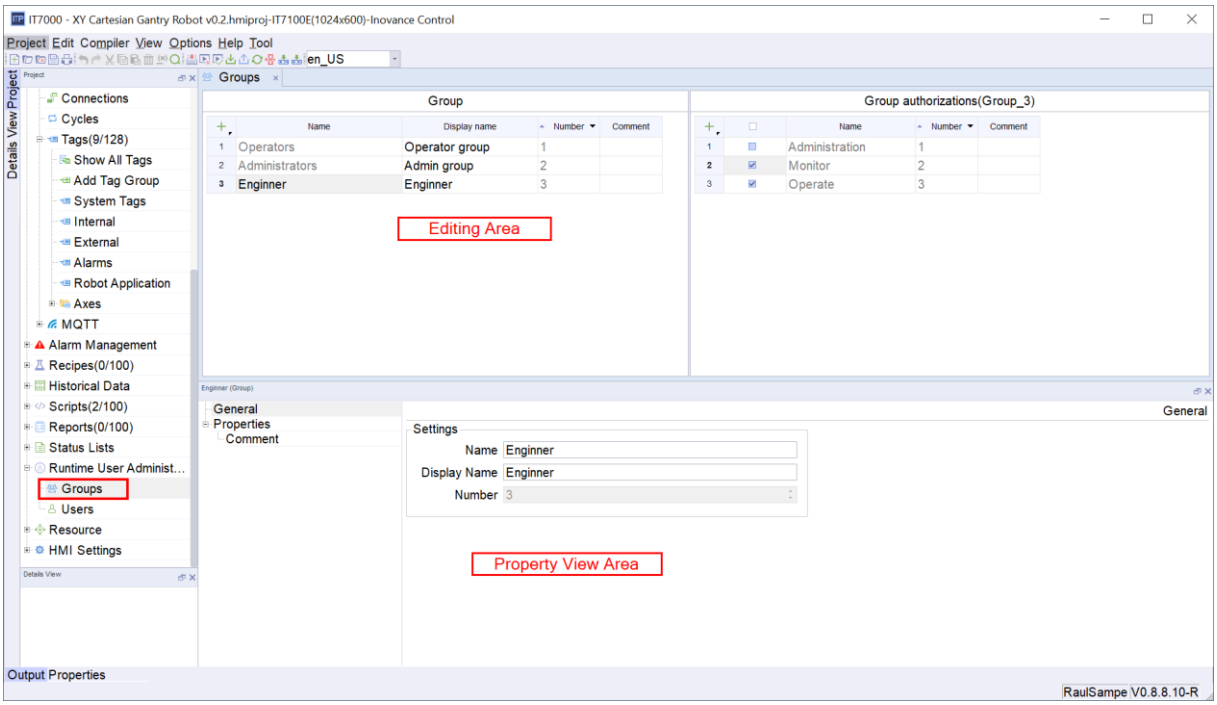
13.3.1 GROUP MANAGEMENT

The list of user groups and their permissions are displayed in the "Groups" management editor. Users can manage user groups and assign permissions to them, up to 50

Groups.

1) Open the "Groups" management editor

As shown in the figure, double-click "Project View Area -> Runtime User Management -> Group" to open the "User Group" editor on the right.



2) Editing area

The "Editing Area" provides functions for displaying and editing user groups and their permissions.

3) Property view area

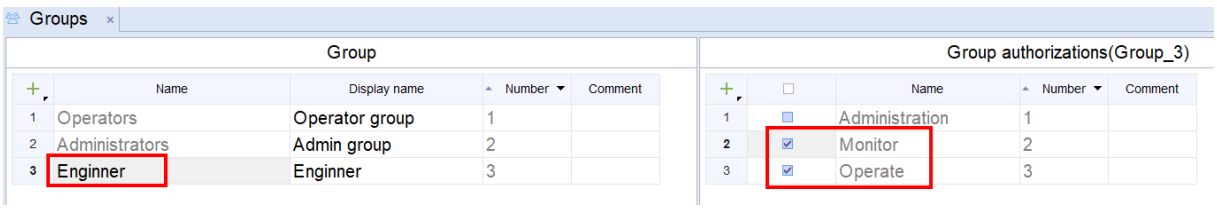
The attribute view provides the same attribute setting function as the table editing area.

4) Configuration instructions

- Groups and group permissions that are grayed out in the "Groups" edit view are automatically generated by the system and cannot be edited by users.
- By default, there are 'Administrator' group and 'Operator' group in the user group.
    - 'Administrator' has all permissions by default, and
    - 'Operator' permissions can be set,

  Up to 32 permissions can be set, and the display name of the group can be used as an internationalized text.

- The checked items in the group authorizations means that the corresponding group has this kind of permission. As shown in the below figure, the group is currently set to 'Enginner', and the 'Operate' and 'Monitor' group authorizations are checked. It means that 'Engineer' group has been granted the permissions of 'Operate' and 'Monitor'

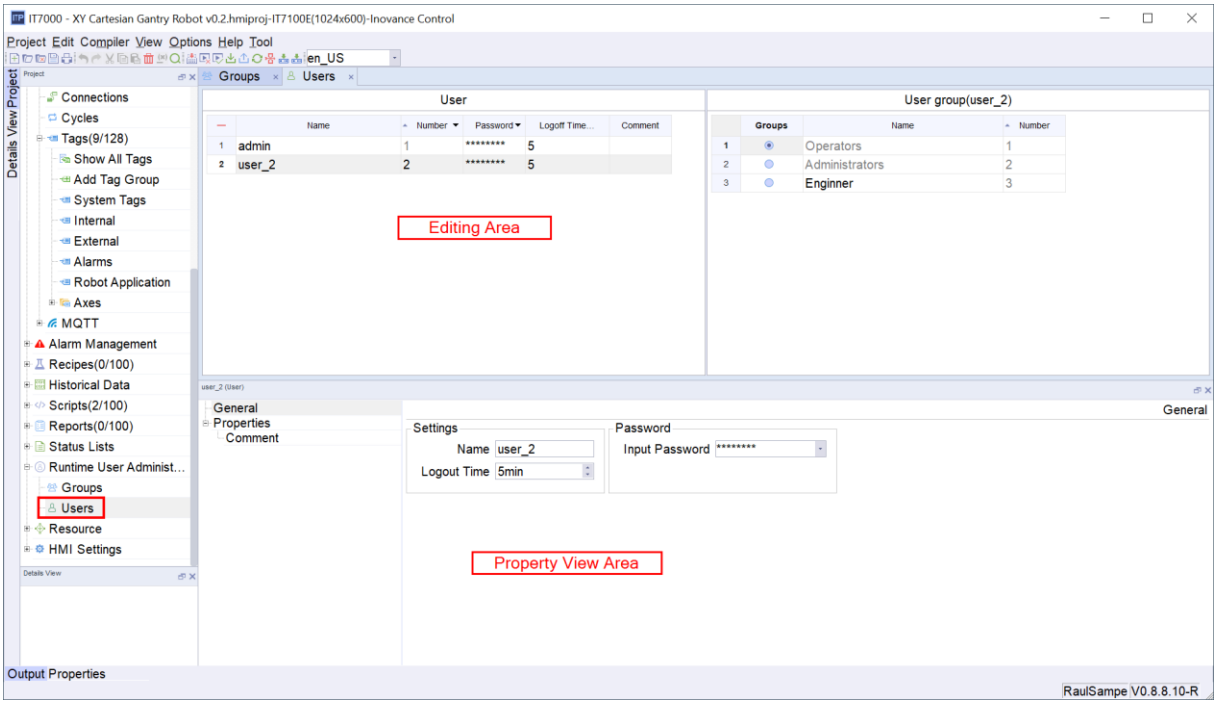


13.3.2 USER MANAGEMENT EDITOR

Users and user groups are listed in a table in the "User" management editor. Here you can manage users and assign them to user groups. You can only create at most 50 users.

1) Open the "User" management editor

As shown in Figure 8-4, double-click "Project View Area -> Runtime User Management -> User" to open the "User" editor shown on the right side of the figure.



2) Editing area

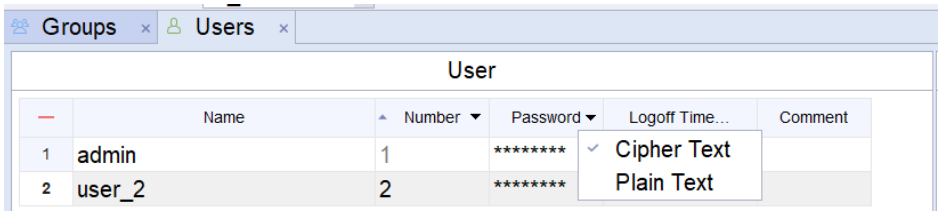
The "Editing Area" provides display and editing functions for users and their groups.

3) Property view area

The attribute view provides the same attribute setting function as the table editing area.

4) Configuration instructions

- Name: It describes the user's name. By default the user "admin" is a member of the administrator group and has the authority of the administrator group (the users and user groups that are grayed out are provided by the system by default. They are not editable).
- Password: the password when the user logs in on the HMI. The password can be viewed as 'plain text' or 'cipher text', and the password length limit is 40 characters long. Only characters [a-zA-z0-9] can be entered. The settings are shown in the below figure.



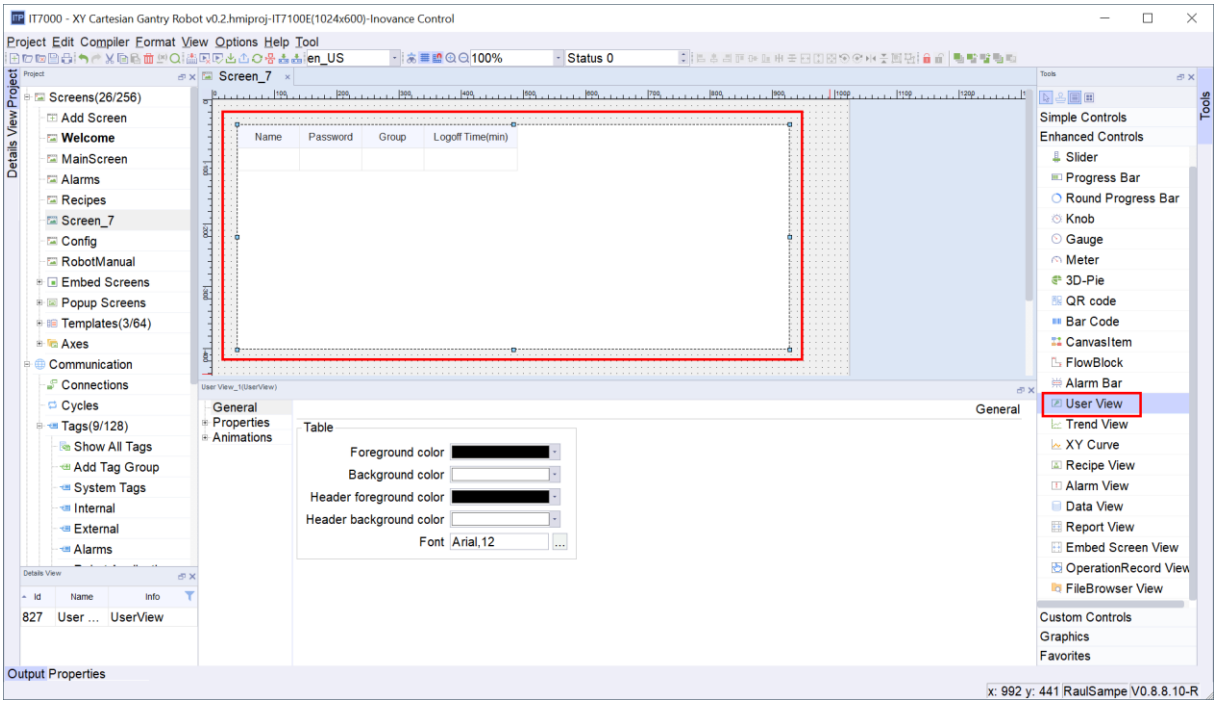
- Logout time (minutes): When the user does not perform any operation (no operation on the mouse and keyboard) within the set time, the current user will be automatically logged out, that is, the user needs to log in again (the system default logout time is 5 minutes). When set to 0, automatic logout is disabled.

13.4 APPLICATION OF USER VIEW IN USER MANAGEMENT

The "User View" is an editing window for managing users in the HMI. Users with "management" permissions can manage all users in the "user view". The restricted users can only manage themselves.

13.4.1 CREATE USER VIEW

As shown in the below figure, click "Enhanced Control -> User View" in the toolbox on the right, and drag it to the screen.

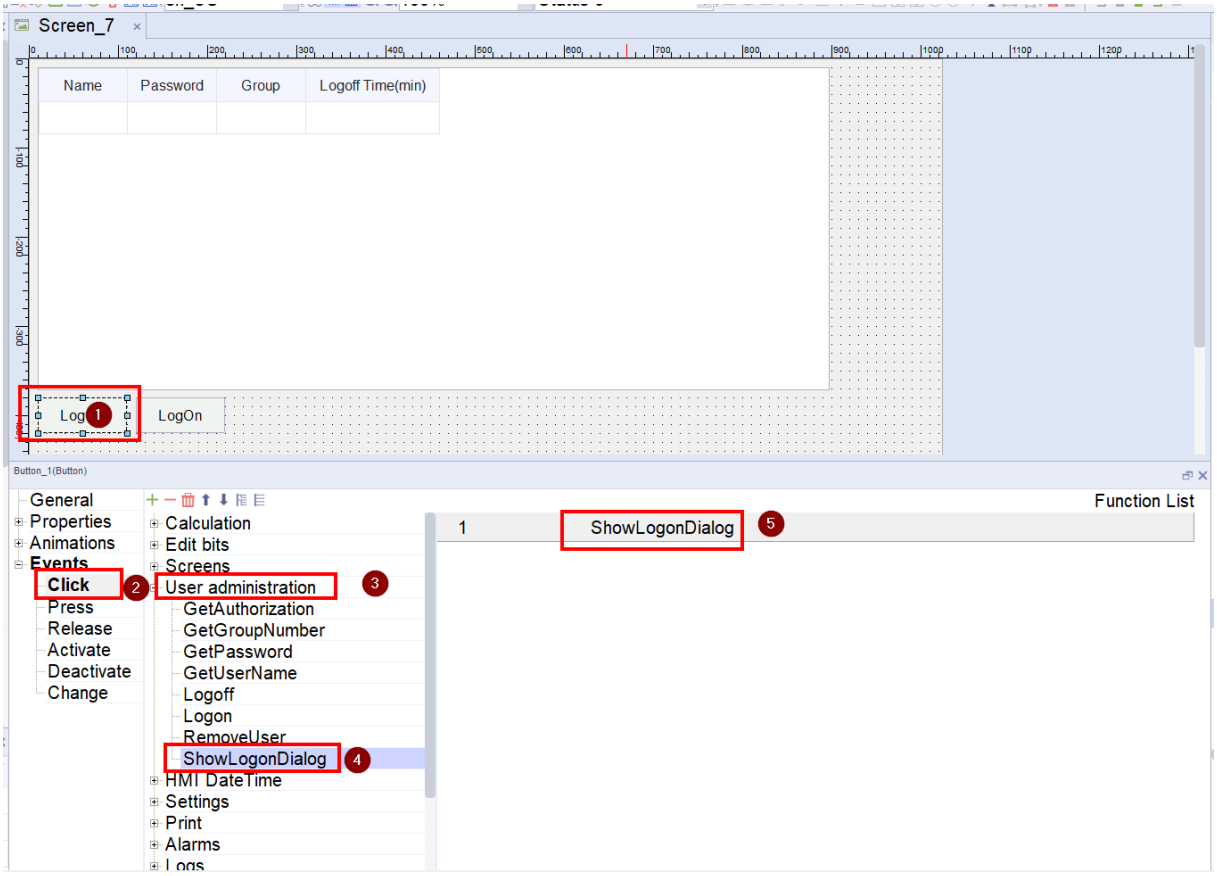


13.4.2 CONFIGURE USER LOGIN/LOGOUT SYSTEM FUNCTIONS

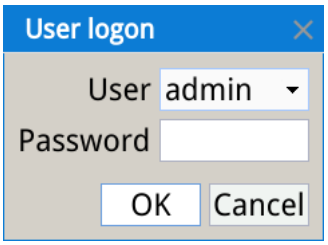
- Select the "Login" button in the project screen, click "Event -> Click -> User Management" in the properties view, and in the expanded system function double-click "ShowLogonDialog" to associate it with the "Logon" button.

- Select the "Logout" button in the project screen, click "Event -> Click -> User Management" in the property view, and in the expanded system function double-click "Logooff" to associate it with the

Logoff" button.



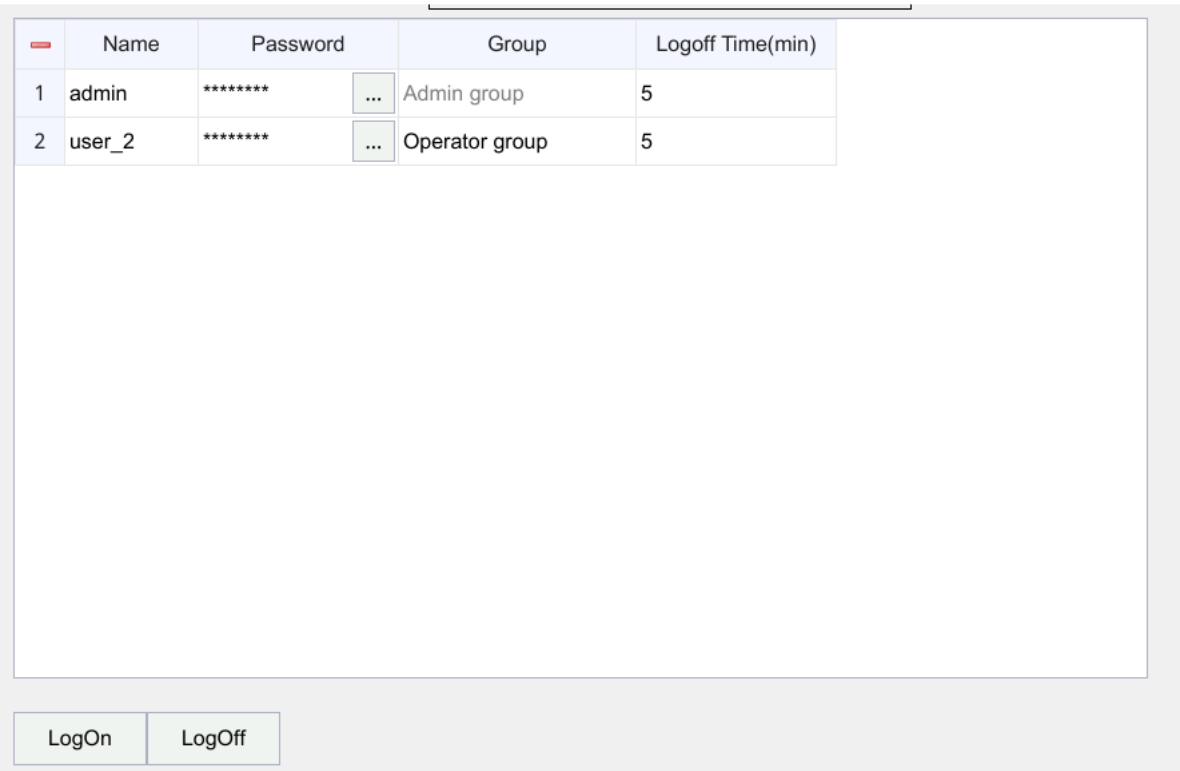
When the button is pressed, the following screen appears where the user credentials must be entered:



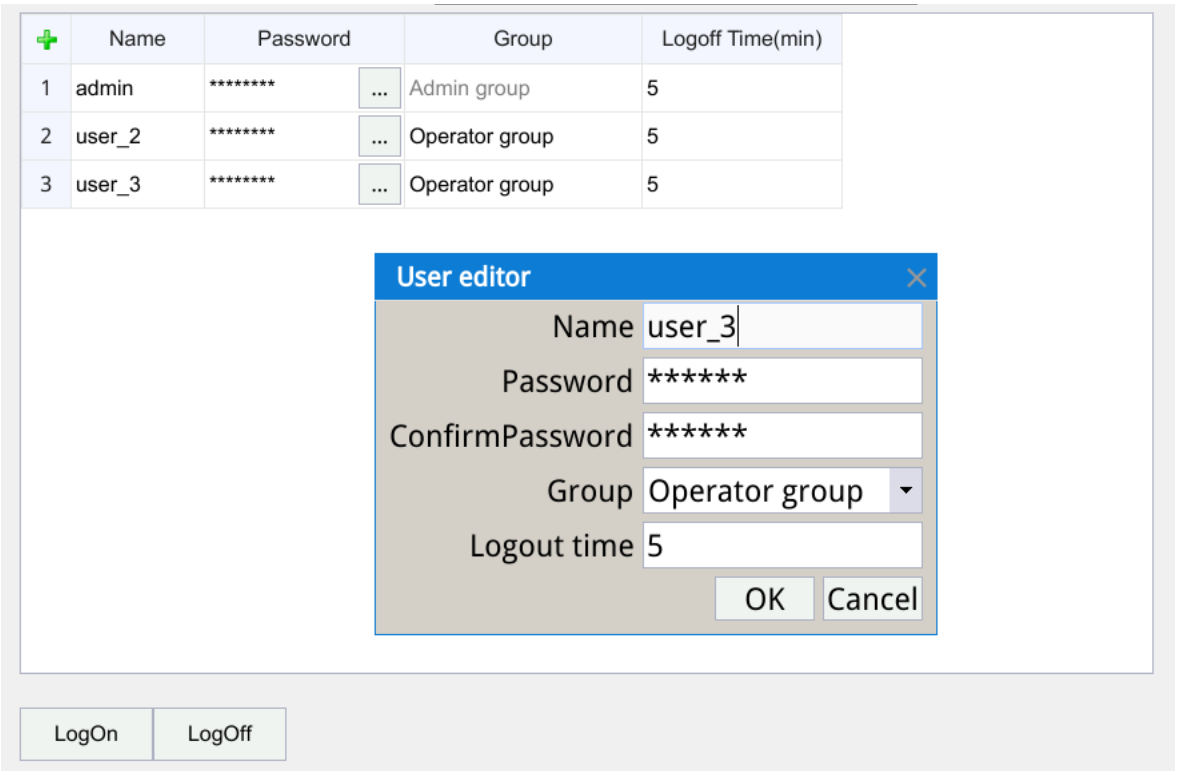
13.4.3 USE USER VIEW IN HMI

After configuring the user management and downloading the project on the screen, it is now possible to modify the users and their permissions from the screen with the user management control. Follow the following procedure to modify the users and their permissions:

1. When the "User View" is initialized, there is no data. Click the "Login" button, in the pop-up "User Login" dialog box, select the designated user to log in. After logging in, you can display your information. After a user with administrative rights logs in, all user information will be displayed.
2. The following figure takes the identity of "admin" as an example, all user information can be displayed in "User View".



3. Double-click the upper left corner of the “User View” in the above figure (the “User View” logged in as a non-administrator does not have this area) to create a new user. (The user name is not restricted, and Chinese user names are supported)
4. You can double-click any cell in the record (non-gray cells, gray means unmodifiable), and an edit window will pop up to make settings, as shown in the figure below.

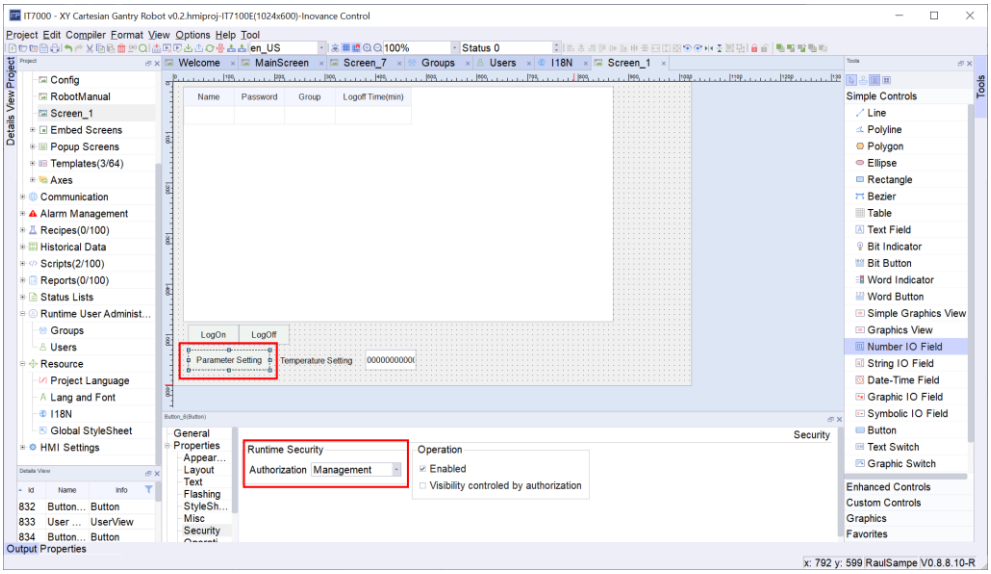


13.5 USER MANAGEMENT APPLICATION

User management allows us to limit access to the components or controls of the screens. The following describes the procedure for assigning user permissions to a project item.

1. Add the "Parameter Setting" button, "Temperature Setting" text field and "Number IO Field" control in "Screen_1", as shown in the figure below.
2. Set the permissions of the "Parameter Setting" button and the "Digital IO Field" control.

   Select the "Parameter Settings" button and enter the "Property View -> Properties -> Security" setting box. Click the drop-down button, select the "management" permission in the drop-down box

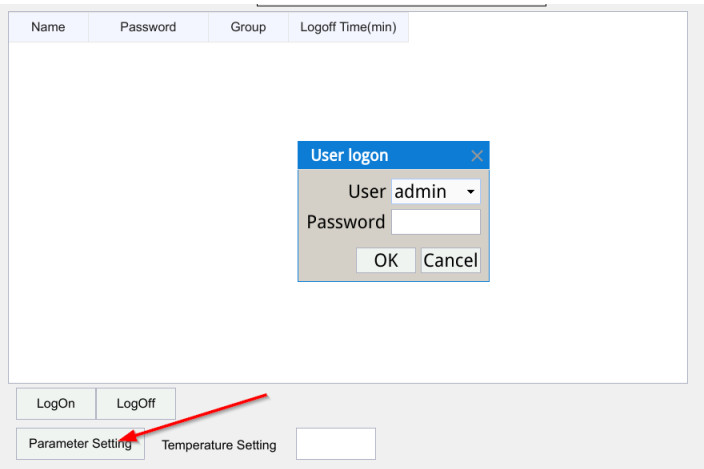


   Perfom the same operation to set the permission of the "digital IO field" control to the "operation" permission

3. Enter HMI

After the project is downloaded to the HMI, if the user clicks on a control configured with permissions without logging in or without authorization, the "Login" dialog box will pop up to control the access permissions.

By default there is no authenticated user, therefore when you try to access these controls, the LonOn screen appears to enter the user's credentials.



14 JAVASSCRIPT

14.1 INTRODUCTION

InoTouchPad provides predefined system functions for common configuration tasks. Users can use them to complete many tasks in the running system, without programming skills. Users can use running scripts to solve more complex problems. The running script has a programming interface and can access part of the project data in the running system.

The use of line scripts is aimed at project designers with JavaScript knowledge.

14.2 SYSTEM FUNCTION PURPOSE

System functions can be configured on all objects that can react to events. Mainly used in function lists and scripts to control the process.

- **Function list**
  The system functions in the function list are processed in order, that is, from the first to the last system function.
- **Script**
  In the script, you can use system functions related to the commands and requirements in the code. In this way, the user can execute the script according to the specified system state.

14.3 ACCESS VARIABLES

The external and internal variables established in the project can be accessed in the script. Variable values can be read or changed at runtime. In addition, you can create in the script local variables, which are used as counters or buffer memory.

■ Example: Use SmartTags('variable name') in the script to access the variables in the project:

```
var a = SmartTags( 'LW 0' );

SmartTags( 'LW 0' ) = 2018;
```

You can insert system functions into the script from the "Script Wizard" or the right-click menu "List all system functions".

■ Example 1: Decrease the value of variable LW 0 by 1 by calling DecreaseValue

```
DecreaseValue(SmartTags( 'LW 0' ),1);
```

■ Example 2: Set a 1s timer, and add 1 to the value of variable LW 0 at a fixed time

```
function add()

{

    var val = SmartTags('LW 0');

    SmartTags('LW 0')=val+1;

}

SetInterval(add, 1000);
```

■ Example 3: Exchange the high and low bytes of variable LW 0, LW 0 is Int16 type

var tag = SmartTags('LW 0');

SmartTags('LW 0')=SwapByte(tag);

For more system functions that support script calls, please refer to the chapter "14.4 System function classification".

14.4 SYSTEM FUNCTION CLASSIFICATION

System functions are predefined functions that can be used to perform many tasks during runtime, even without any programming knowledge, such as:

- Calculate, for example, increase the value of a variable by a specific value or variable value.
- Logging functions, such as starting process value logging.
- Settings, such as changing the PLC or setting bits in the PLC.
- Alarm, for example, an alarm is issued after another user logs in.

14.4.1 SCREEN-RELATED SYSTEM FUNCTIONS

ID	System function		Description	Script call
1	**ActivateScreen**(ScreenName)		Switch to the specified screen by screen name	Support, for example: ActivateScreen('picture _1');
	Parameter	ScreenName		
2	**ActivateScreenByNumber**(ScreenId)		Switch to the specified screen by screen number	Support, for example: ActivateScreenByNumber (SmartTags('Variable_1'));
	Parameter	ScreenId		
3	**ActivatePreviousScreen**		Switch to the previously displayed screen	Support, for example: ActivatePreviousScreen();
4	**ShowPopup**		Display a pop-up screen at the specified location	Support, for example: ShowPopup('picture _2',100,200);
	Parameter	ScreenName	Pop-up screen name	
		posX	Display position X, centered by default. Constant / variable	
		posY	Display position Y, centered by default. Constant / variable	
5	**HidePopup**		Hide pop-up screen	Support, for example: HidePopup('picture _2');
	Parameter	ScreenName	Pop-up screen name	

14.4.2 CALCULATE RELATED SYSTEM FUNCTIONS

ID	System function		Description	Script call
1	**DecreaseValue** (Tag, Value, Reset)		Tag = Tag – Value	Support, for example: DecreaseValue(SmartTags('variable _1'),1,0);
	Parameter	Tag(InOut)	Variable, the default value is empty	
		Value(In)	Constant/variable, default value 1	
		Reset	Reset, select yes/no, default no, when you select yes,	

			After the variable is reduced to the lower limit, it will be reset to the upper limit of the variable Limit.	
2	**IncreaseValue** (Tag, Value，Reset)		Tag = Tag + Value	Support, for example: IncreaseValue(SmartTags('variable_1'),1,1);
	Parameter	Tag(InOut)	Variable, the default value is empty	
		Value(In)	Constant/variable, default value 1	
		Reset	Reset, select yes/no, default no, when you select yes, After the variable is added to the upper limit, it will be reset to the lower limit.	
3	**InverseLinearScaling** (X, Y, b, a)		X = (Y-b) / a Variables X and Y cannot be the same	Support, for example: InverseLinearScaling(SmartTags('Variable_1'),SmartTags('Variable _2'),2,2);
	Parameter	X(Out)	Variable, the default value is empty	
		Y(In)	Variable, the default value is empty	
		b(In)	Constant/variable, default value 0	
		a(In)	Constant/variable, default value 1	
4	**LinearScaling**(Y, X, a, b)		Y = (a *X) + b Variables X and Y cannot be the same	Support, for example: LinearScaling(SmartTags('variable_1'),3,SmartTags('Variable _2'),3);
	Parameter	X(Out)	Variable, the default value is empty	
		a(In)	Constant/variable, default value 1	
		X(In)	Variable, the default value is empty	
		b(In)	Constant/variable, default value 0	
5	**SetValue** (Tag, Value)		Tag = Value	Support, for example: SetValue(SmartTags('Variable_1'),22);
	Parameter	Tag(InOut)	Variable, the default value is empty	
		Value(In)	Constant/variable, default value 0	
6	**Random** (Tag)		Take the random value of the upper and lower limits of the variable	Support, for example: Random(SmartTags('Variable _1'));
	Parameter	Tag(Out)	Variable, the default value is empty	

14.4.3 BIT MANIPULATION RELATED SYSTEM FUNCTIONS

ID	System function		Description	Script call
	InvertBit (Tag)		Invert the value of a given "Bool" type variable	Support, for example: InvertBit(SmartTags('Variable_1'));
		Tag(InOut)	Variable, the default value is empty	

	InvertBitInTag (Tag, Bit)		Invert the bits in a given variable	Support, for example: InvertBitInTag(SmartTags('Variable _2'),1,0);
		Tag(InOut)	Variable, the default value is empty	
		Bit(In)	Tag, default value: 0, range (0-15)	
	ResetBit (Tag)		Set the value of the "Bool" type variable to 0	Support, for example: ResetBit(SmartTags('Variable_1'));
		Tag(InOut)	Variable, the default value is empty	
	ResetBitInTag(Tag, Bit)		Set a bit in the given variable to 0	Support, for example: ResetBitInTag(SmartTags('Variable_2'),0);
		Tag(InOut)	Variable, the default value is empty	
		Bit(In)	Tag, default value: 0, range (0-15)	
	SetBit(Tag)		Set the value of "Bool" type variable to 1	Support, for example: SetBit(SmartTags('Variable_1'));
		Tag(InOut)	Variable, the default value is empty	
	SetBitInTag(Tag, Bit)		Set a bit in the given variable to 1	Support, for example: SetBitInTag(SmartTags('Variable_2'),0);
		Tag(InOut)	Variable, the default value is empty	
		Bit(In)	Tag, default value: 0, range (0-15)	

14.4.4 TIMER FUNCTION

ID	System function		Description	Script call

1	**SetInterval**(expression,value)		Set the timer and execute after the specified time is reached Corresponding function (execute multiple times in a loop)	Support, for example:
				function add()
				{
				var val= SmartTags('variable _1');
				SmartTags('variable_1')=val+1;
				}
				SetInterval(add , 1000);

	parameter	expression	The name of the function being executed	
	parameter	value	Specify the timing time, the unit is ms	
2	**ClearInterval**(timer)		Clear timer, paired with SetInterval	Support, for example: var timer=SetInterval(add , 1000); ClearInterval(timer);
	parameter	Timer	Timer created with SetInterval	

3	**SetTimeout**(expression,value)		Set the timer and execute after the specified time is reached Corresponding function (only executed once)	Support, for example:
				function add()
				{
				var val= SmartTags('variable_1');
				SmartTags('variable_1')=val+1;
				}
				SetTimeout (add , 1000);

	parameter	expression	The name of the function being executed	
	parameter	value	Specify the timing time, unit ms	
4	**ClearTimeout**(timer)		Clear timer, paired with SetTimeout	Support, for example:

	parameter	value	Timer created with SetTimeout	var timer= SetTimeout (add , 1000); ClearTimeout (timer);

14.5 JAVA SCRIPT EXAMPLES

14.5.1 DATA CHANGE

```
var arr=new Array(10);
var i;

{//
arr[0]=SmartTags("LW 100");
arr[1]=SmartTags("LW 101");
arr[2]=SmartTags("LW 102");
arr[3]=SmartTags("LW 103");
arr[4]=SmartTags("LW 104");
arr[5]=SmartTags("LW 105");
arr[6]=SmartTags("LW 106");
arr[7]=SmartTags("LW 107");
arr[8]=SmartTags("LW 108");
arr[9]=SmartTags("LW 109");
}
```

```
	for(i=0;i<10;i++)
	{
		arr[i]=Math.floor(Math.random()*100);//0~100 random number
	}
```

```
{//
SmartTags("LW 100")=arr[0];
SmartTags("LW 101")=arr[1];
SmartTags("LW 102")=arr[2];
SmartTags("LW 103")=arr[3];
SmartTags("LW 104")=arr[4];
SmartTags("LW 105")=arr[5];
SmartTags("LW 106")=arr[6];
SmartTags("LW 107")=arr[7];
SmartTags("LW 108")=arr[8];
SmartTags("LW 109")=arr[9];
}
```

14.5.2 MATHEMATICAL FUNCTIONS

```
var  value1=SmartTags("LW 20");
var  value2=SmartTags("LW 24");
var  value3=SmartTags("LW 28");
var  value4=SmartTags("LW 32");
var  value5=SmartTags("LW 36");
var  value6=SmartTags("LW 40");
var  value7=SmartTags("LW 44");
var  value8=SmartTags("LW 50");
var  value9=SmartTags("LW 52");
```

```
var  value10=SmartTags("LW 56");
var  value11=SmartTags("LW 58");
var  value12=SmartTags("LW 62");
var  value13=SmartTags("LW 64");

SmartTags("LW 22")=Math.abs(value1);       // Return absolute value
SmartTags("LW 26")=Math.ceil(value2);       // Round up
SmartTags("LW 30")=Math.floor(value3);      // Round down
SmartTags("LW 34")=Math.round(value4);    // Round a number to the nearest
whole number
SmartTags("LW 38")=Math.exp(value5);       // Returns the exponent of e
SmartTags("LW 42")=Math.log(value6);        // Return the natural logarithm
of the number (base e)
SmartTags("LW 46")=Math.sqrt(value7);       // Returns the square root of
the number
SmartTags("LW 48")=Math.random();           // Returns a random number
between 0 and 1

var  max = Math.max(value8 , value9);        // Returns the higher of x and
y
var  min  = Math.min(value10 , value11);     // Returns the lower of x and y
var  pow = Math.pow(value12 , value13);     // Returns x to the power of y
SmartTags("LW 54")= max;
SmartTags("LW 60")=min;
SmartTags("LW 66")=pow;


var  radian=SmartTags("LW 0")*Math.PI/180;

SmartTags("LW 2")=Math.sin(radian);       // Sine function
SmartTags("LW 4")=Math.cos(radian);      // Cosine function
SmartTags("LW 6")=Math.tan(radian);      // Tangent function

var  value1=SmartTags("LW 8");
var  value2=SmartTags("LW 10");
var  value3=SmartTags("LW 12");

SmartTags("LW 14")=Math.asin(value2)*180/Math.PI;      // Returns the arc
sine of a number
SmartTags("LW 16")=Math.acos(value1)*180/Math.PI;     // Returns the arc
cosine of a number
SmartTags("LW 18")=Math.atan(value3)*180/Math.PI;    // Returns the arc
tangent of x with a value between -PI/2 and PI/2 radians
```

14.5.3 STRING MANIPULATION

```
var str1=String( SmartTags("Variable_1") );
var str2=SmartTags("Variable_2");
var str3=SmartTags("Variable_3");
var str4=SmartTags("Variable_4");
var str5=SmartTags("Variable_5");
var str6=SmartTags("Variable_7");
var str7=SmartTags("Variable_8");
var str8=SmartTags("Variable_9");

 SmartTags("Variable_10")=str1.indexOf(str3);         //Retrieve the
position of the string str3 in str1, return the position of the first
occurrence of the retrieved string, return -1 if not found
 SmartTags("Variable_11")=str1.lastIndexOf(str4);  //Retrieve the position
of the string str4 in str1, return the position of the first occurrence of
the retrieved string, return -1 if not found
```

```
 SmartTags("Variable_6")=str1.match(str5);           //Match str5 in str1,
match to display the string, no return empty
 SmartTags("Variable_12")=str1.search(str6);         //Match str6 in str1,
match to the position where the string is displayed, without returning -1

var temp=str1.replace(str7 , str8);                   //In str1, replace
str7 with str8
SmartTags("Variable_14")=temp;

var  index3=SmartTags("Variable_21");
temp=str1.substr( index3);                  // From the start index number
to the end, extract the string
SmartTags("Variable_20")=temp;


SmartTags("Variable_22")=str1.toLowerCase();  //Convert string to lowercase
SmartTags("Variable_23")=str1.toUpperCase();  //Convert string to uppercase

SmartTags("Variable_24")=str1.concat(str2);           //Connection string
```

14.5.4 STRING CONVERSION

```
var  mynumber=new Number(SmartTags('Variable_1'));

SmartTags("Variable_2")=mynumber.toString();// Convert a number to a
string, use the specified base (default base 10), display in decimal, base
range 2~36
SmartTags("Variable_3")=mynumber.toString(2);// Convert number to string,
base=2, binary display
SmartTags("Variable_4")=mynumber.toString(16);// Convert a number to a
string, base=16, display in hexadecimal

var  mynumber=Number(SmartTags("Variable_5"));

SmartTags("Variable_6")=mynumber.toFixed(1);   // Convert a number to a
character string, the result has a specified number of digits after the
decimal point, and a specified 1 decimal place is displayed
SmartTags("Variable_7")=mynumber.toFixed(2);   // Convert the number to a
character string, the result has a number with a specified number of digits
after the decimal point, and specify 2 decimal places for display

var  mynumber=Number(SmartTags("Variable_8"));

SmartTags("Variable_9")=mynumber.toPrecision(1);// Format the number to a
specified length, and specify a scientific notation with a length of 1
SmartTags("Variable_10")=mynumber.toPrecision(2);// Format the number to a
specified length, and the specified length is 2 scientific notation
```

15 E-MAIL NOTIFICATIONS CONFIGURATION

The IT7000 allows you to configure the sending of notifications via email. These emails can contain information from the TAGs configured in the HMI project. This way you can send information about the process that the IT7000 is monitoring.

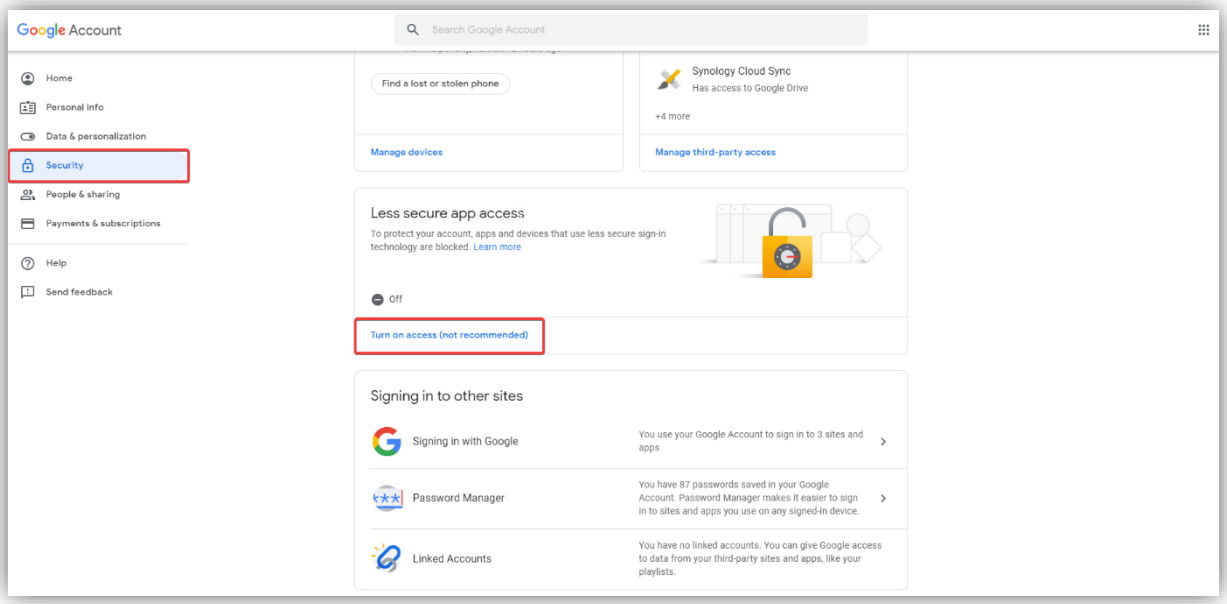
15.1 SET UP SMTP SEVER USING GMAIL

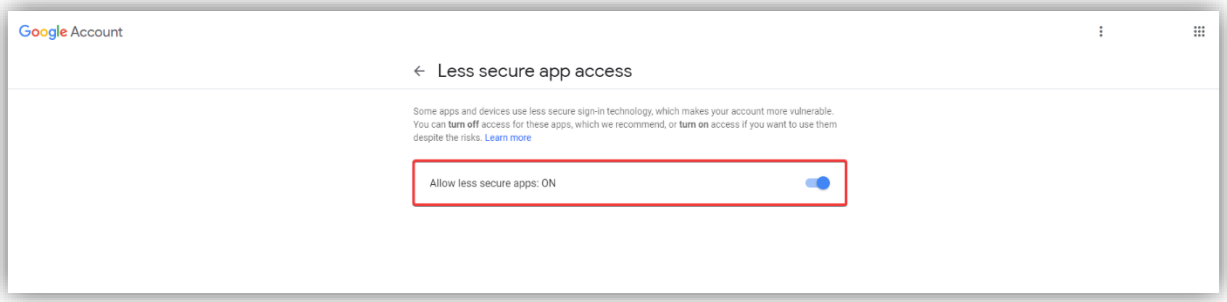
In order to configure this functionality, it is necessary to be able to access an SMTP server to send the emails. This section shows how to configure a Gmail account to be able to send the messages.

Authorize access to your Gmail account

Sign in to your Gmail account

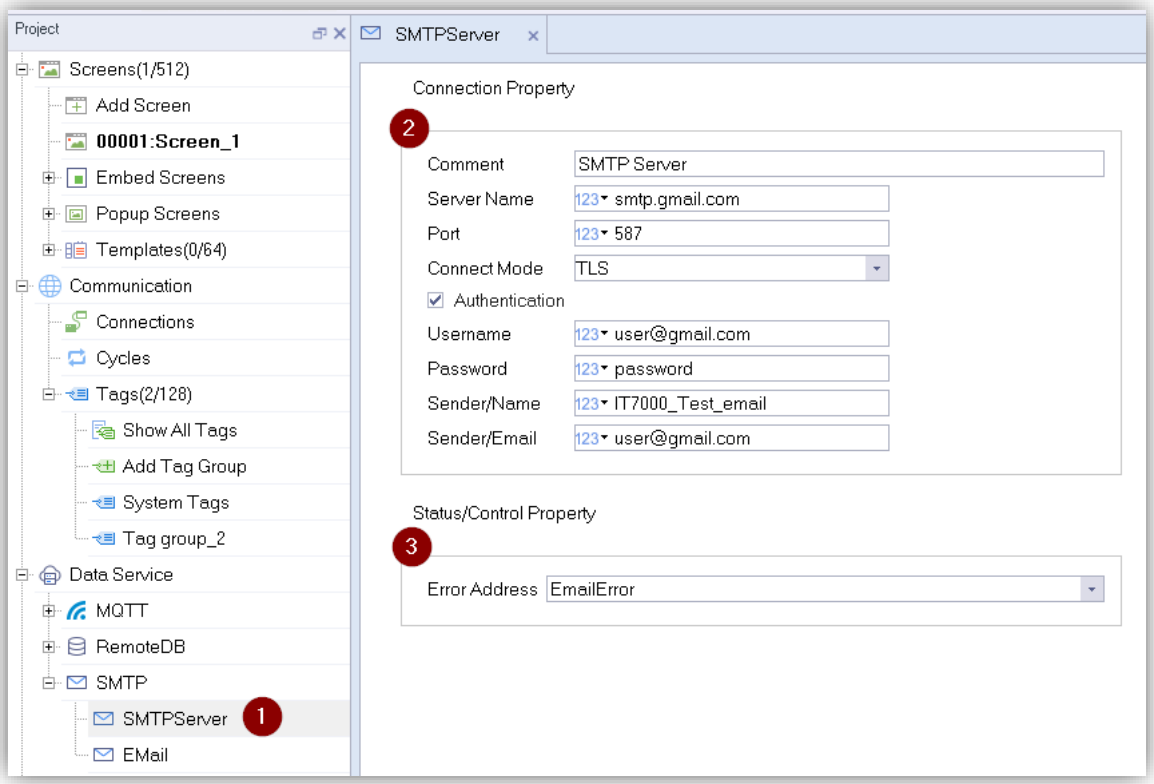
Go to **Security** > **Less secure app access**. Click **Turn on access**. Less secure app access is not available for accounts with 2-step verification enabled.





Set up SMTP server

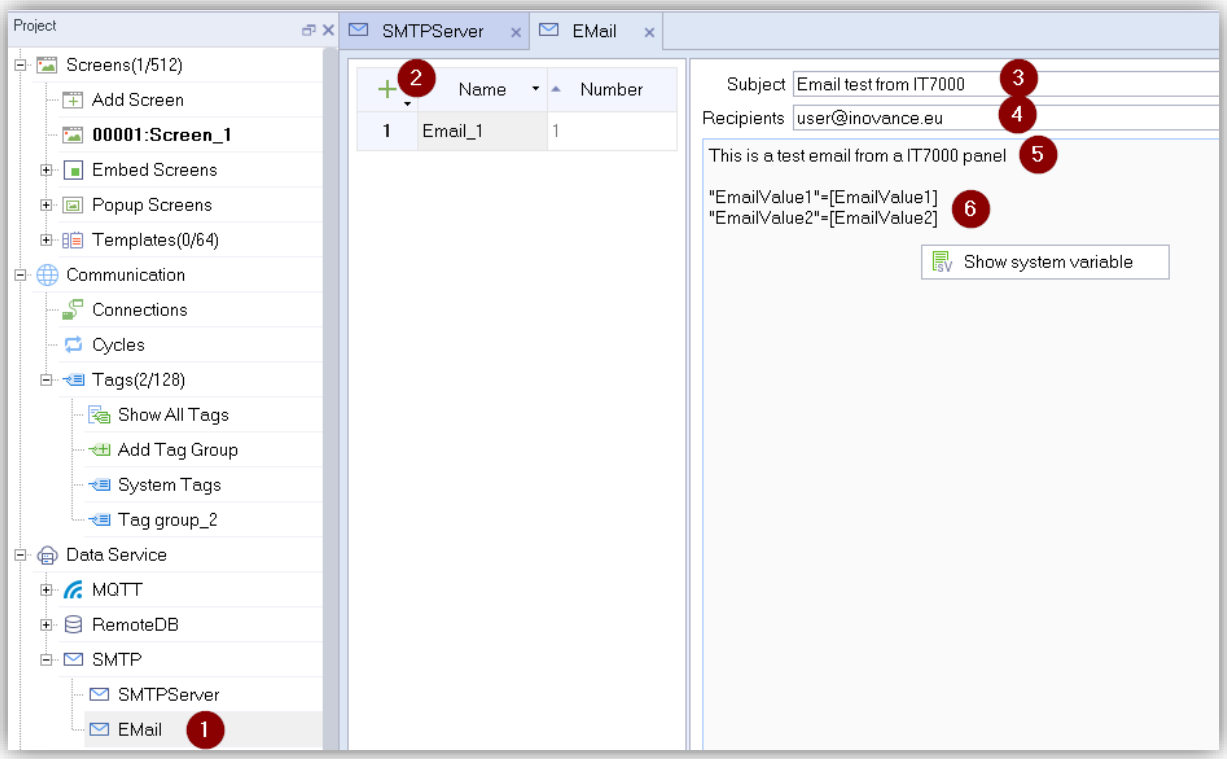
Configure the IT7000 STMP server as shown in the following image. The username and password must be the ones used to access to your Gmail account.



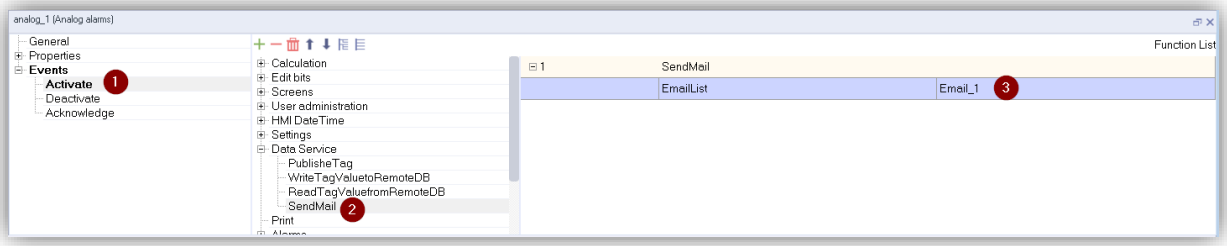
15.2 SET UP EMAIL NOTIFICATIONS

To create an email notification, follow the steps below:

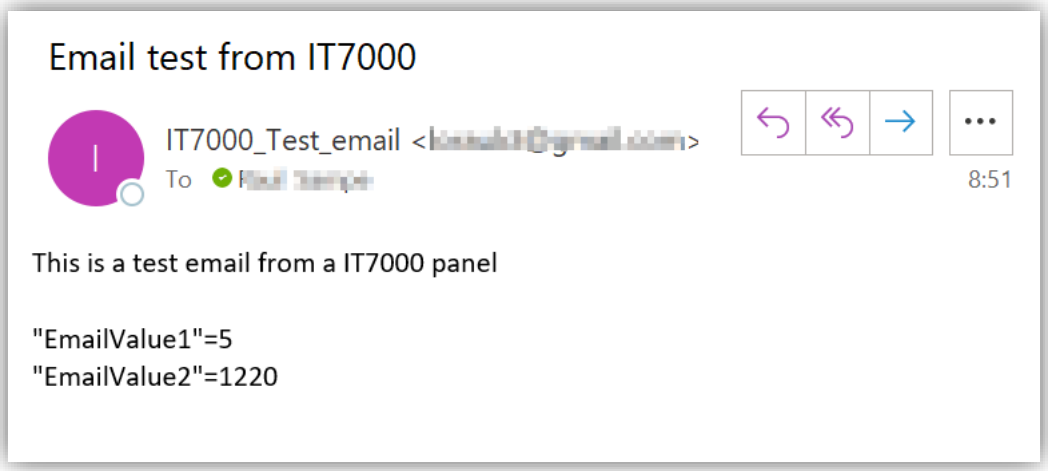
1. Open the email notification screen
2. Create a new notification
3. Add email subject
4. Add the recipient of the email
5. Add the notification text
6. Add process variables or TAGs. To add process values, you can write the name of the TAG in brackets or by clicking with the right mouse button, the option "Show system variables" appears that allows us to add any TAG defined in the project.



The screen can send an email notification through the events of a button, an alarm, a timer, ...



This is the mail received from the IT7000 when the values of the TAGs "EmailValue1" and "EmailValue2" are 5 and 1220 respectively.



16 AM600 TIPS

The AM600 is a Inovance's centralized motion controller with built-in PLC functionality. The IT7000 display can communicate with the AM600 controller through the Modbus RTU, Modbus TCP and Qlink TCP protocols.

This section describes some tips to facilitate communication between the IT7000 and the AM600.

16.1 FLOATING POINT NUMBER COMMUNICATION

The IT7000 display communicates with the AM600 controller through the Modbus protocol. Modbus itself does not define a floating point data type but it is widely accepted that it implements 32-bit floating point data using the IEEE-754 standard.

A floating point variable is stored according to the IEEE 754 standard which it can transfered as 4 bytes, but not by the REAL_TO_WORD conversion. This can be done using UNION data unit type:

Create the union DUT:

```
TYPE MODBUS_RealToDWord :
UNION
    dwDword : DWORD;
    rReal : REAL;
END_UNION
END_TYPE
```

Create a program to transfer the application variable to Modbus variable:

```
PROGRAM PLC_PRG
VAR
    diRobotXPosition AT %MD501 : MODBUS_RealToDWord;
END_VAR

diRobotXPosition.rReal := 12.2020;
```

Setup a Float TAG in the InoTouchPad:

	Name	Number	Connection Id	Data type	Length	Array count	Address
1	Robot - X Position	670	Connection_1	Float	4	1	MD 501

16.2 MODBUS ADDRESSING

When the AM600 controller acts as a Modbus slave and we want to access the variables of the InoProShop project, the address of the variables must be defined so that it can be configured in the TAGs of the InoTouchPad.

Direct address is used to define the address of the variables. A direct address, also called fixed address or direct variable, has a direct mapping to a specific address of the PLC. The address information includes the variable storage location in the CPU, storage size, and storage position offset.

The syntax of direct address is as follow:

Syntax: %<store area prefix><size prefix><number>|.<number>

- The programming system supports the following three **storage area prefixes**:
    1. **I**: input, physical input, sensor
    2. **Q**: output, physical output, executor
    3. **M**: storage location
- The programming system supports the following **size prefixes**:
    1. **X**: bit, 1 bit
    2. **B**: byte, 1 byte
    3. **W**: word, 1 word
    4. **D**: double word, 4 bytes (double byte)

The first number indicates the offset address of the memory prefix corresponding to the variable. The number following "." indicates the specific bit after the address offset when the variable is of the boolean type.

Example:

%QX7.5 output area with 7-byte offset, sixth bit (bit 5)

%QX17 output area with 17-byte offset

%IW215 input area with 215-word offset

%MD48 memory area with 48-double word offset

iVar AT %IW10: WORD; //The variable iVar is of the word type and maps to the location with 10-word offset

in the input area.

NOTE When a variable with the X-type size prefix indicates the boolean data type, the offset address needs to be precise to bits.

NOTE The size prefix matches with the data type. A variable with the B-type size prefix must be declared as a 1-byte data type, such as BYTE, SINT, and USINT. A variable with the W-type size prefix must be declared as a 1-word data type, such as WORD, INT, and UINT. A variable with the D-type size prefix must be declared as a double-word data type, such as DWORD, DINT, and UDINT.

Addressing by bit	Addressing by byte	Addressing by word	Addressing by Dword		Addressing by bit	Addressing by byte	Addressing by word	Addressing by Dword
QX0.0	QB0	QW0	QD0		MX0.0	MB0	MW0	MD0
QX0.1					MX0.1			
QX0.2					MX0.2			
QX0.3					MX0.3			

QX0.4					MX0.4			
QX0.5					MX0.5			
QX0.6					MX0.6			
QX0.7					MX0.7			
QX1.0	QB1				MX1.0	MB1		
QX1.1					MX1.1			
QX1.2					MX1.2			
QX1.3					MX1.3			
QX1.4					MX1.4			
QX1.5					MX1.5			
QX1.6					MX1.6			
QX1.7					MX1.7			
QX2.0	QB2	QW1			MX2.0	MB2	MW1	
QX2.1					MX2.1			
QX2.2					MX2.2			
QX2.3					MX2.3			
QX2.4					MX2.4			
QX2.5					MX2.5			
QX2.6					MX2.6			
QX2.7					MX2.7			
QX3.0	QB3				MX3.0	MB3		
QX3.1					MX3.1			
QX3.2					MX3.2			
QX3.3					MX3.3			
QX3.4					MX3.4			
QX3.5					MX3.5			
QX3.6					MX3.6			
QX3.7					MX3.7			
QX4.0	QB4	QW2	QD1		MX4.0	MB4	MW2	MD1
QX4.1					MX4.1			

16.3 ADDRESS STORAGE AREAS

The direct address storage area varies depending on different PLCs. PLC data is not retained upon power failure for the %I and %Q areas, but is retained for the %M area.

The AM600, AM610, AM401, and AM402 programming systems provide the 128 KB (byte) input area (I area), 128 KB (byte) output area (Q area), and 512 KB storage area (M area). The first 480 KB of the storage area can be used directly, whereas the last 32 KB are used by the system, mainly as soft elements, and cannot be used directly by users. During programming, you can directly access addresses, or define variables and map the variables to addresses for indirect access. The following table lists the storage areas and address ranges.

Area	Function	Size	Address Range
I area (%I), 128 KB	Area used by users	64 Kwords	%IW0 to %IW65535
Q area (%Q), 128 KB	Area used by users	64 Kwords	%QW0 to %QW65535
M area (%M), 512 KB	Area used by users	240 Kwords	%MW0 to %MW245759
	SD element	10,000 words	%MW245760 to %MW255759
	SM element	10,000 bytes	%MB511520 to %MB521519
	Reserved	2768 bytes	%MB521520 to %MB524287