

Help Manual for the Programming Page

1	Terms and Definitions	3
2	Pre-requirements	3
3	UI initialization	4
4	Programming mode.....	4
4.1	Programmer Login	4
4.2	Programming Page Overview	6
4.3	uController Configuration	6
4.4	IO Configuration.....	9
4.5	Logout Button	10
4.6	System Time Synchronization.....	11
4.7	Project Control panel	11
4.8	Project Explorer area	12
4.8.1	Action buttons	12
4.8.2	Projects.....	13
4.8.3	Project Controls	15
4.8.4	Python Libraries.....	20
4.9	Project IOs Definition Section	22
4.10	Activation/Deactivation IO states.....	28
4.11	Project Configuration section	29
4.11.1	Action buttons	29
4.11.2	Routine Entries	34
4.11.3	Routine Control Buttons.....	34
4.11.4	CON Routines editor.....	35
4.11.5	Trends Configuration	37
5	Reordering & misconfiguration cases	40
5.1	Preconditions	40
5.2	Cases Analysis	40
5.2.1	Bad or no communication	40

5.2.2	Swapping IOs of same type	40
5.2.3	Swapping IOs of different type	41
5.2.4	Replacement of IO of same type.....	41
5.2.5	Replacement of IO of different type.....	41
5.2.6	Adding IOs at the end	41
5.2.7	Adding IOs in between.....	41
5.2.8	Missing IOs.....	41
5.3	Unsuccessful reordering	42
6	Programming/Run pages synchronization	43
7	Active Sessions.....	45
8	Graphics	45
8.1	SCR and CON Routines Graphic Pages	45
8.2	Graphic Page Control Buttons	47
8.3	Left Side Menu.....	48
8.4	Right Side Menu	50
8.4.1	Copy, Paste, Cut, Delete, Duplicate.....	52
8.4.2	Magnify/ Minify.....	52
8.4.3	Up, Down, Left, Right.....	52
8.4.4	Bring To Front/ Send To Back.....	52
8.4.5	Selection Type.....	52
8.4.6	Link Page.....	53
8.4.7	Group Objects.....	55
8.5	Object Info Card.....	56
8.6	Objects Categories	58
8.6.1	Embedded Objects	58
8.6.2	Interactive Objects	59
8.6.3	Imported Objects	75
8.7	Mini Map.....	75
9	Python modules	76

1 Terms and Definitions

IDLE	Python Integrated Development Environment
IO	Any of the following devices: AIN5A, AOUT8A, CIO5A, DIN8A, DDIN5A, DOUT8A, ROUT5A.
UI	User Interface.
Trend	A set of a channel's recorded values.
Log	A Trend's recorded value with its timestamp.
uC	Microcontroller. The main chip of the CON64A device
LOC	Line Of Code

2 Pre-requirements

The currently supported browsers are:

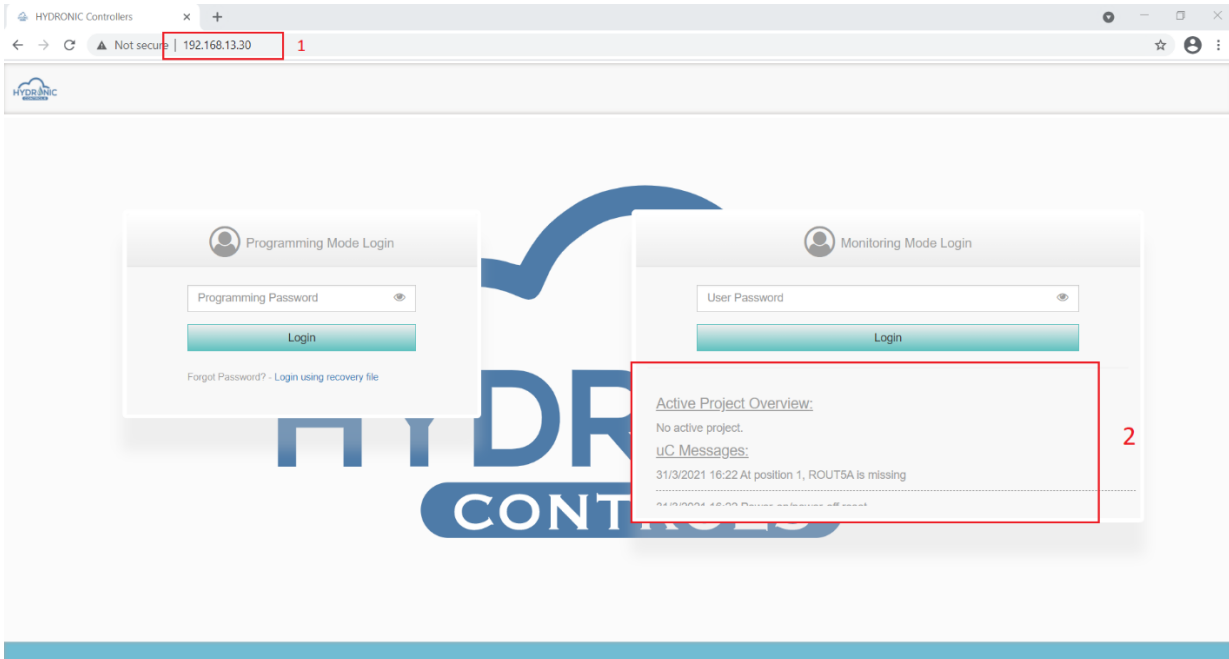
- Mozilla Firefox
- Chrome
- Edge (the chromium version)

Notes:

1. As of Google Chrome update to version 83.0.4103.97, the checkboxes are displayed blue while color combination. They can be restored to their original settings (in greyed color) by navigating to the following address <chrome://flags/#form-controls-refresh> and disabling the flag "Web Platform Controls updated UI".
2. By default, Google Chrome blocks pop-ups from automatically showing up on your screen. To only allow pop-ups from the IP of the CON64A you can do the following:
 - 1) Navigate to your browser's setting e.g. <chrome://settings/>
 - 2) Under "Privacy and security", click Site settings
 - 3) Click Pop-ups and redirects
 - 4) Click the Add button next to "Allow" label
 - 5) Enter the IP of the CON64A e.g. <http://192.168.10.10>
3. The [Active Sessions section](#) describes a feature that needs at least the Chrome version 81.0.4044.122 in order to work correctly.

3 UI initialization

To access the User Interface (UI), the user should type in the URL field of the browser, the IP of the CON64A. e.g. 192.168.13.10 (see 1 in image below). Upon page load, the initial screen displayed contains two login areas. One for the Programming page (Programming Mode Login) and another one for the Run page (Monitoring mode Login). The user can see information related to an active project (if exists) and messages logged from the uC (see 2 in image below).



Index page of Hydronic Controller

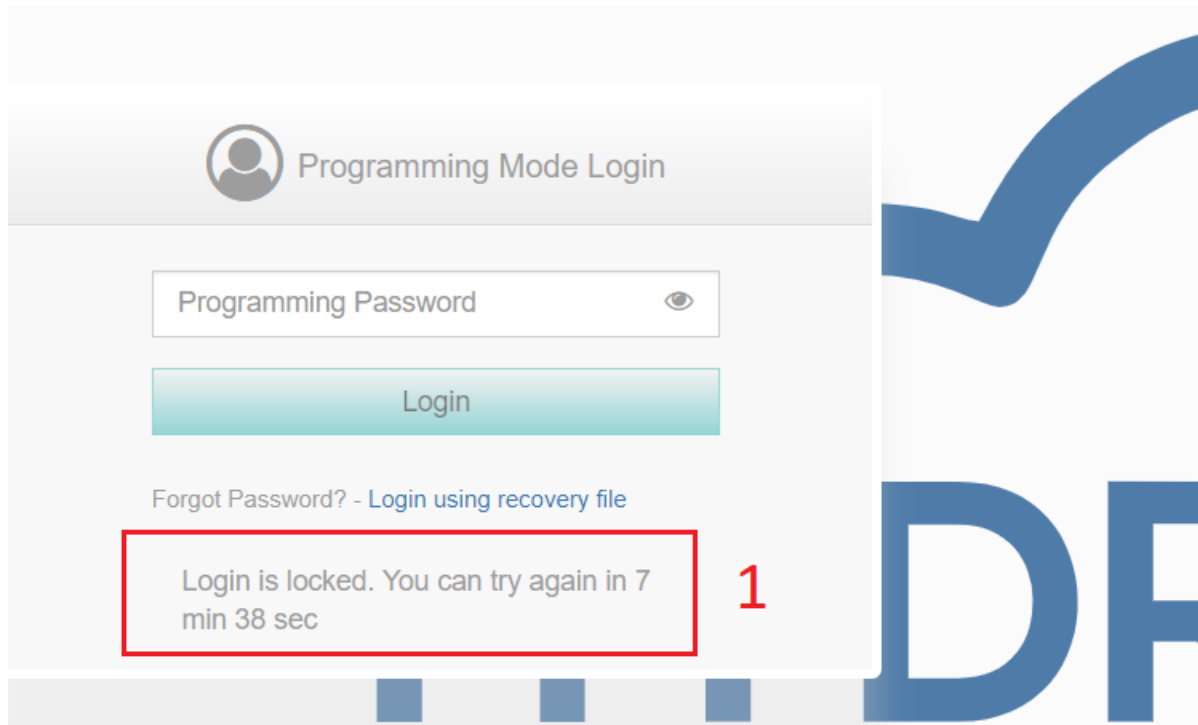
4 Programming mode

This mode is accessible only from the system programmer and in this section the main functionalities of the UI and the user interactions are described.

4.1 Programmer Login

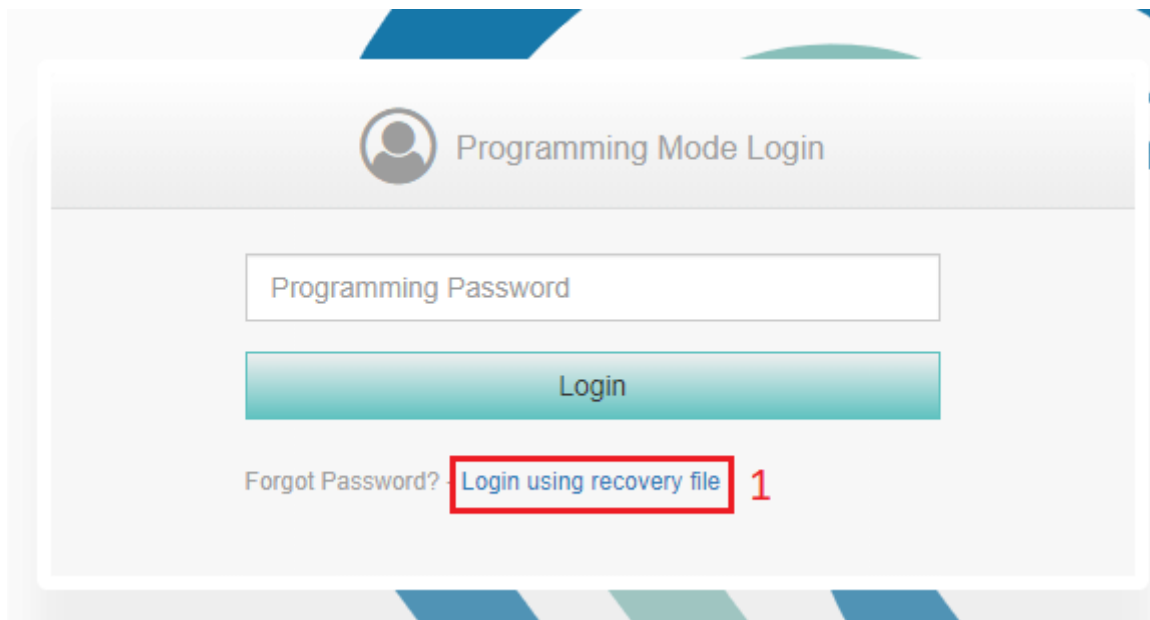
To log in as a Programmer to the UI the default password is 'changeme'.

To avoid attacks to the system the login attempts are limited to 5. After this threshold the login functionality is disabled for 10 minutes (See 1 of image below). The programmer will not be able to login from any browser during this time period.



Programming mode locked login

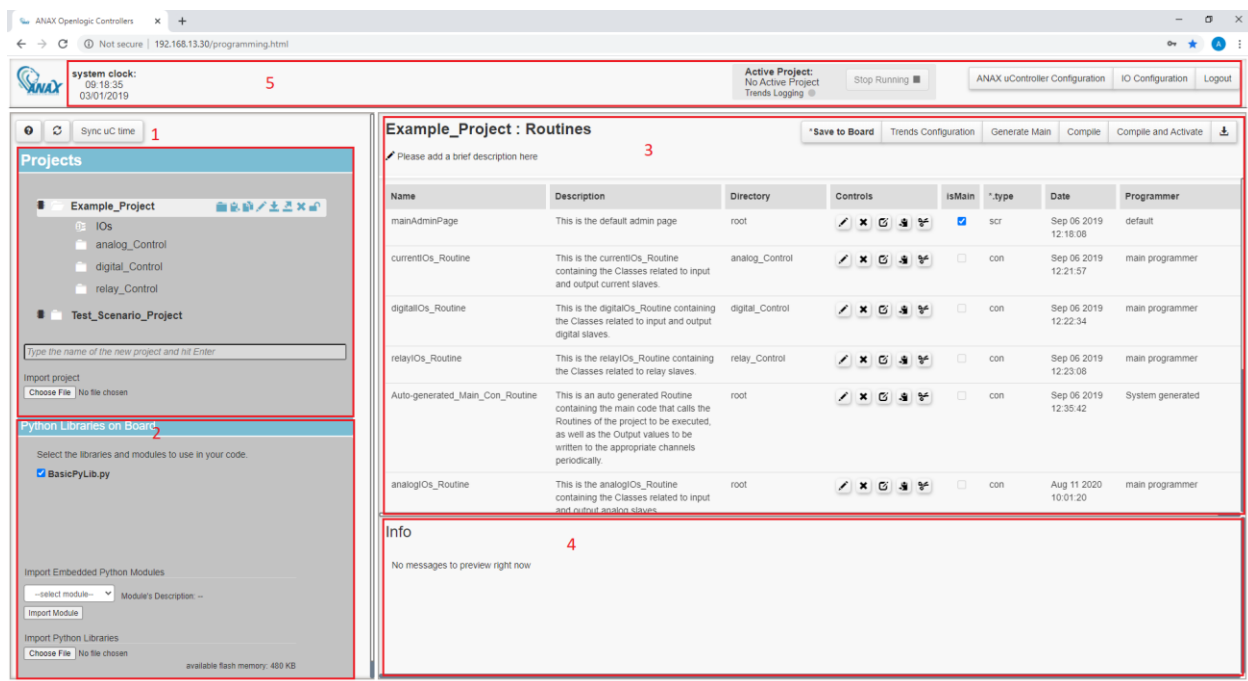
An additional mechanism in case the programmer has forgotten his password allows him to login by using a recovery file (See 1 of image below) that can be generated from the [uController Configuration](#). This file has to be generated in advance (when the programmer is still able to login) and stored somewhere for future use.



Login using recovery file

4.2 Programming Page Overview

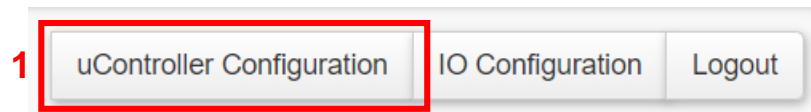
The programming page is divided as follows: the project explorer section (See 1 of Image below) where all the user projects are listed, a python libraries section (See 2 of Image below) that lists any available python libraries existing on the CON64A device, a routine section (See 3 of Image below) that shows all the available routines of a project along with interaction routine related buttons, an Info section (See 4 of Image below) to display messages during programming and some useful ucontroller related buttons (See 5 of Image below).



Programming Page Overview

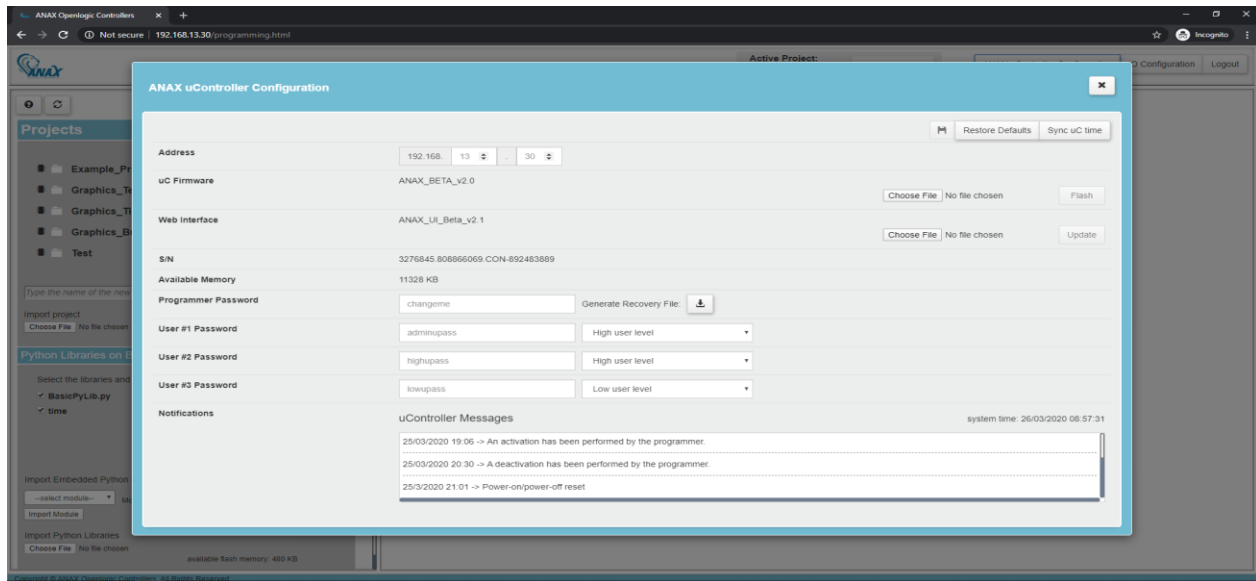
4.3 uController Configuration

The programmer can access the uController configuration by clicking the corresponding button on the top right corner of the programming page (See 1 of image below).



Hydronic uController Configuration Button

A new modal will open with all the available configuration options.

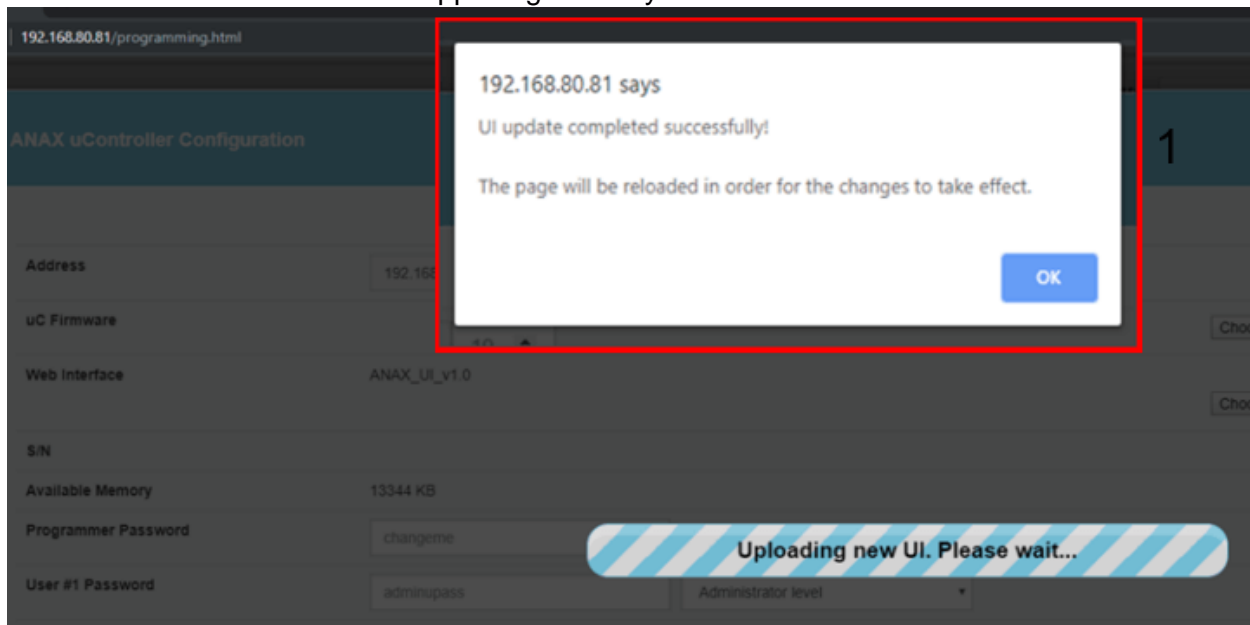


uController Configuration Modal of Hydronic controller

The programmer can modify the default IP 192.168.10.10 of the CON64A device to a new desired one.

He can also update the CON64A Firmware as well as the UI Firmware from the corresponding options. In both cases he has to select the corresponding file for the update. For the CON64A firmware a *.bin file is provided and a *.zip file for the UI. By uploading the appropriate file, the procedure is initiated after clicking the “Flash” button for the CON64A firmware and the “Update” button for the UI.

The update procedure is automated and throws appropriate messages during the update so that the user is informed of what is happening in the system.



UI update procedure completes successfully

The new version of UI is displayed in the “Web Interface” field of uController Configuration (See uController Configuration Image above) e.g. HYDRONIC_UI_v2.8 . The programmer’s personal settings such as IP and passwords are not affected from such a procedure.

The Serial number (“S/N”) of the CON64A device is also displayed for identification purposes.

The programmer can also check the “Available Memory” on the uController. The available memory is checked upon login/save to board actions and when the onboard memory drops below 4000KB the programmer is informed accordingly. Additionally, in case the onboard memory is below 2000KB, further save to the board is disabled for safety reasons.

From the uController Configuration modal the programmer has also the ability to change passwords for the predefined users and assign the desired access levels.

Administrator Level is the highest Level for the users that may join to the run page with all the supported functionality of the Run page available, then the High user Level is and Low user Level may be used in order to limit functionalities from the Run page.

As the Programmer is the only user that can edit all the passwords an option for access to the programming page is given in case, he has forgotten his password. Upon the very first login the programmer shall download the recovery file related to the specific uC, by clicking the button next to “Generate Recovery File” label, and store it for future login with this file.

The “Restore Defaults” button is used to reset the uC to its default settings, which will take place either after you log out of the current session or after restarting the power of the uC. If the IP of the uC was changed previously, for the programmer now to be able to initialize the UI, he needs to enter the default IP in the URL field of the browser.

The “Sync uC time” button is used to sync the uC’s time to the programmer’s system time.

Finally, real-time information and notifications related to the uC are logged in the uController Messages area.

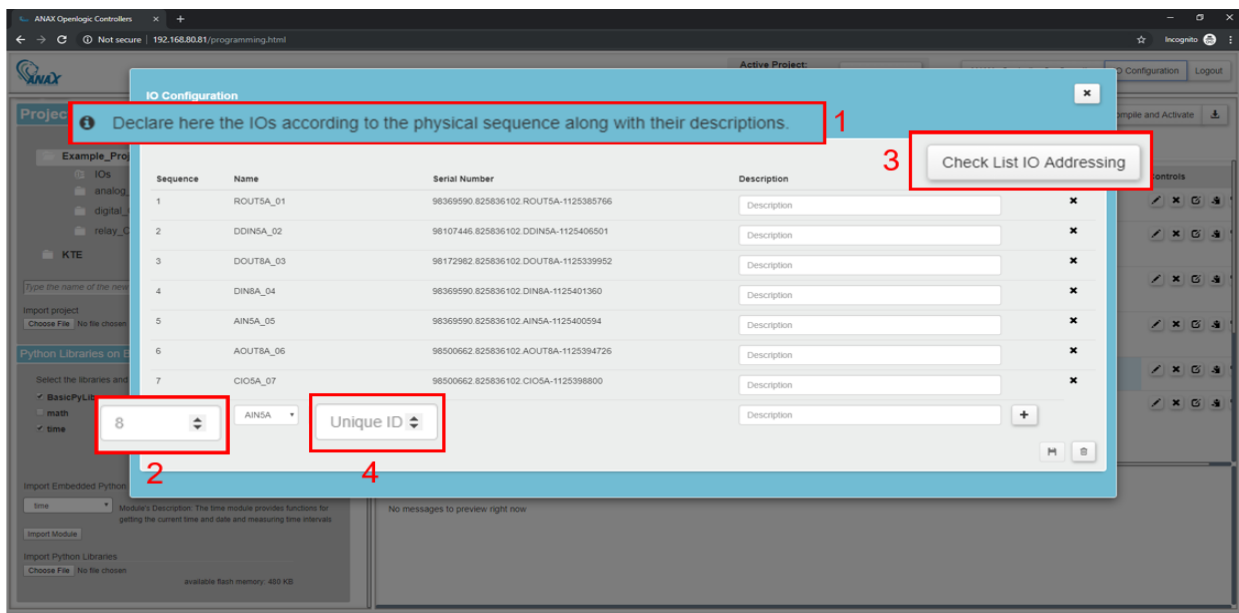
4.4 IO Configuration

“IO Configuration button” is located on the top right corner of the Programming Page too.

In the IO Configuration modal the programmer shall declare the physical IOs he is going to use, according to their physical sequence to the communication channel along with a description of his choice (See 1 in image below). The sequence field is auto-completed, with an incremental number depending on the number of the already added IOs to the list (See 2 of image below). This number though may be manually given in case the IO to be added is not at the latest available position. The input number in this case shall be the exact position of the new IO and the rest IOs to the list will be pushed further down.

The programmer shall also fill the ‘Unique ID’ (See 4 of image below) which is a unique numeric identifier from 1 to 64, to address the IO devices during communication.

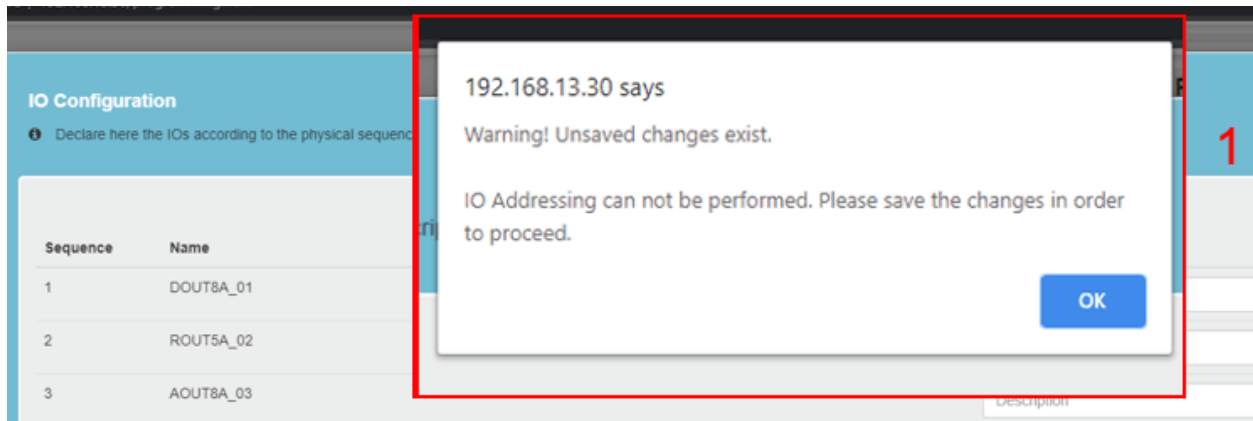
After the programmer has declared all the IOs connected to the CON64A and saved this list he can execute the IO Addressing procedure. This procedure will try to identify the declared IOs to the communication channel and retrieve their Serial Numbers. The procedure starts by clicking the button ‘Check List IO Addressing’ (See 3 of image below).



Sequence	Name	Serial Number	Description
1	ROUTSA_01	98369590.825836102.ROUTSA-1125385766	Description
2	DOINSA_02	98107446.825836102.DOINSA-1125406501	Description
3	DOUTBA_03	98172982.825836102.DOUTBA-1125339952	Description
4	DINBA_04	98369590.825836102.DINBA-1125401360	Description
5	AINSA_05	98369590.825836102.AINSA-1125400594	Description
6	AOUTBA_06	98500662.825836102.AOUTBA-1125394726	Description
7	CIOA_07	98500662.825836102.CIOA-1125396800	Description

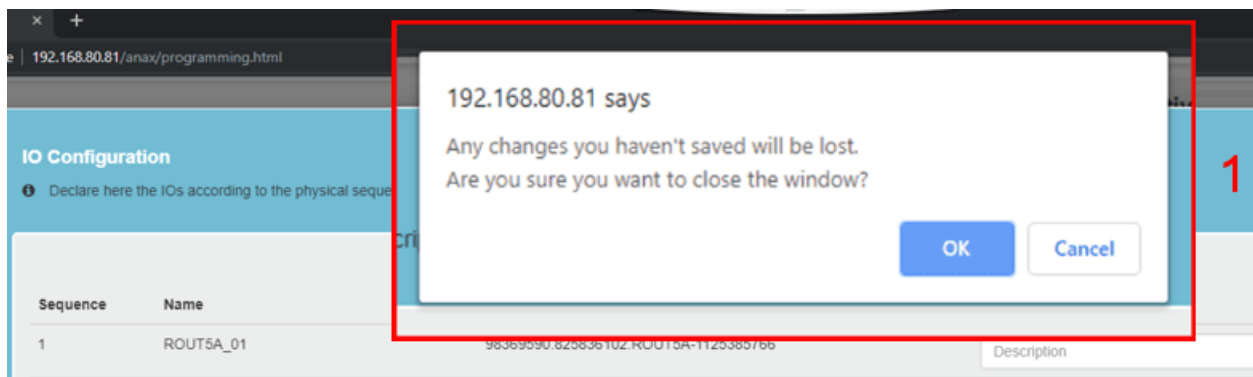
IO Configuration modal of Hydronic Controller

In case of unsaved changes in IO Configuration, the IO Addressing cannot be performed unless the programmer saves the changes first (See 1 of image below)



IO Configuration modal of Hydronic controller

Upon closing the IO Configuration window, the user is alerted if unsaved changes exist. He has the ability either to cancel this procedure and save his changes or close the window leaving the unsaved changes to be rejected (See 1 of image below)



IO Configuration modal of Hydronic controller

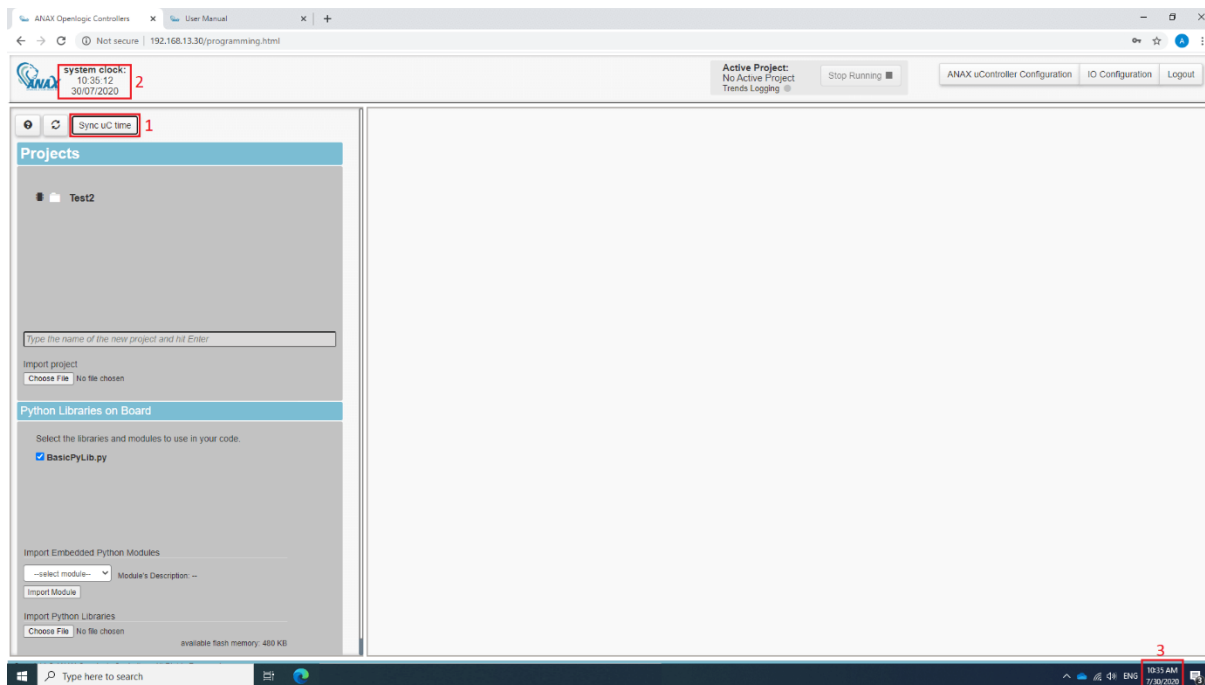
4.5 Logout Button

Next to the "IO Configuration" button there is also the "Logout" button. By clicking the "Logout" button the user will be redirected to the initial screen and the active session of the run page will end.

4.6 System Time Synchronization

On the top left corner of the programming page and next to the Logo the system's time (MCU time) is shown (See 2 of image below).

The time is read every 10 minutes from the Board and can be synchronized to the local computer's clock by clicking the "Sync uC time" (See 1 of image below). . This button will read the local computer's clock and send it to the Board. When the CON64A is not powered the time is not updated and has to be synced again when accessing the UI.

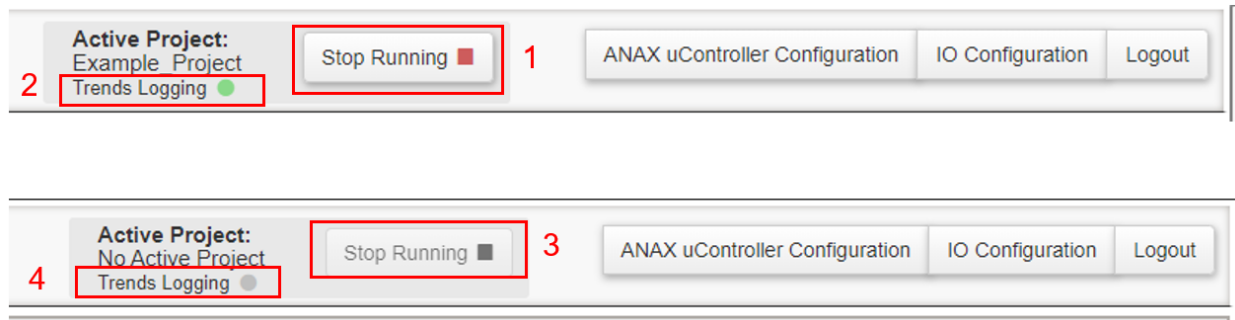


System Time Synchronization

4.7 Project Control panel

On the top right corner of the programming page, a small control panel for the Active Project was created. The programmer can see the name of an active project, if exists, and has the ability to deactivate it. When a project is active, the stop icon is enabled and red. (See 1 in image below). A blinking led shows when the system's Trends Logging ([explained in 4.10 Trends Configuration](#)) is enabled for the Active project (See 2 of image below).

If no Active project exists, the deactivate button is disabled (See 3 in image below) and the Trends Logging led is grey and inactive (See 4 in image below).

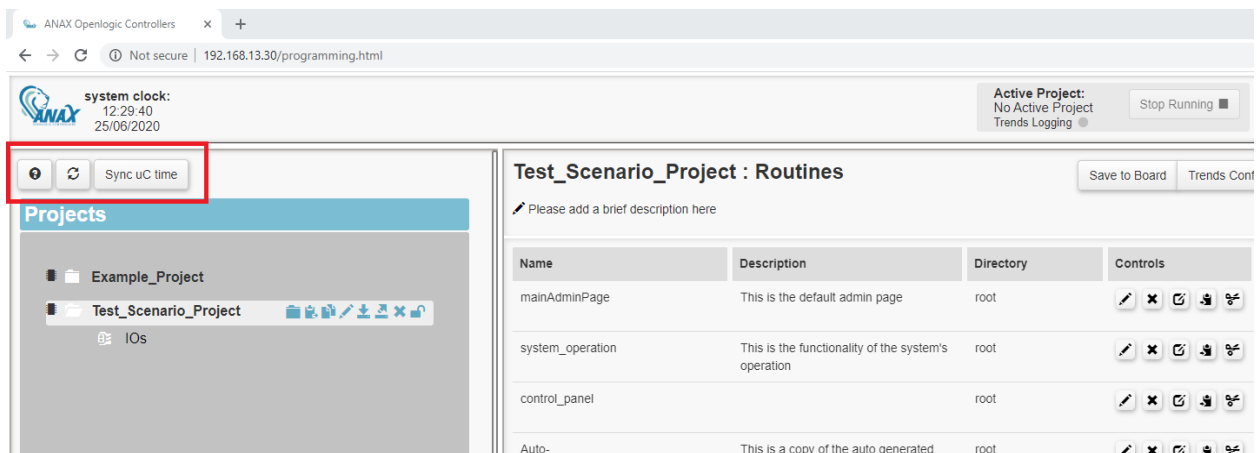


Project control panel of Hydronic controller

4.8 Project Explorer area

4.8.1 Action buttons

On the top of the project explorer area the programmer can see three buttons.



Help/Restore data/Sync uC time buttons

The first one is a “Help” button. By clicking this button, a new tab will open containing this exact help manual for the Programming page.

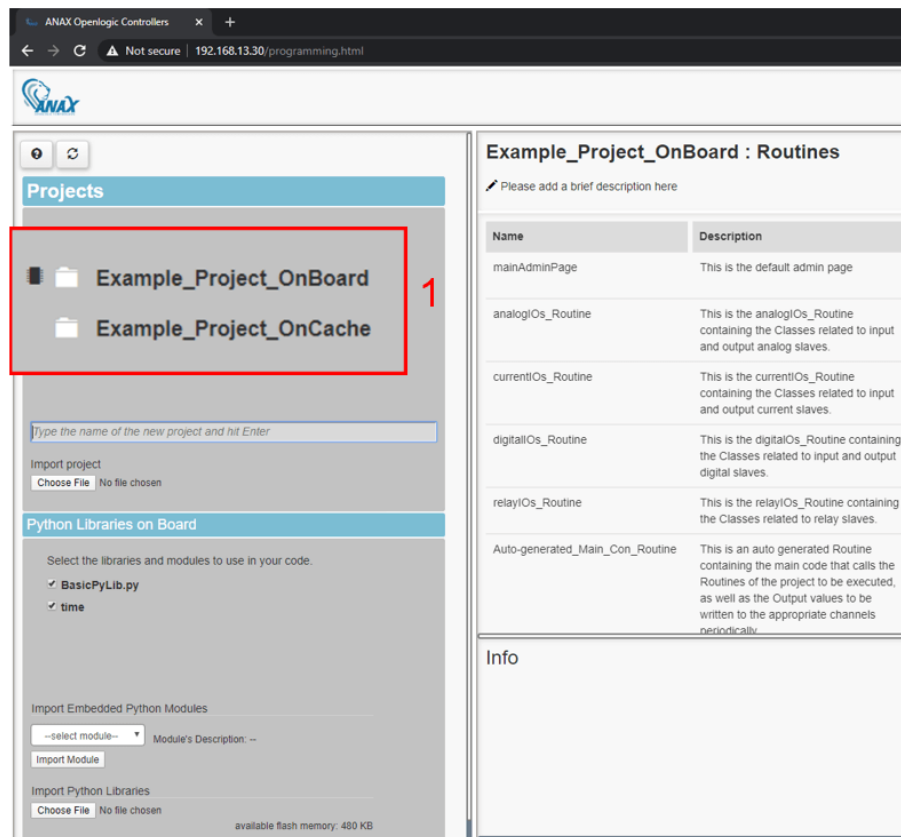
By clicking the second button, the programmer can restore data that are saved on the board. This functionality can prove useful if the programmer deletes a project from the browser's cache by mistake.

The third button is the “Sync uC time” button, which was already explained in the System Time Synchronization section.

4.8.2 Projects

The Projects area lists all the projects currently available both on the uController memory and the Browser's cache.

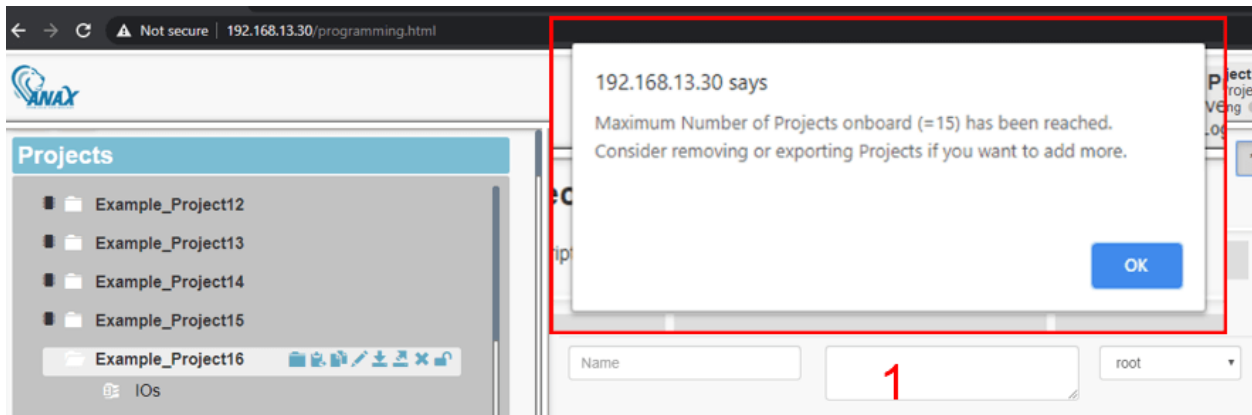
Up to 15 projects can be stored on the board and an unlimited number of projects can exist on the browser's cache, but only one project and only if it is stored on board, can be active at a time. A chip icon next to a project's name indicates that the project is saved to the board. Projects without the icon are only stored in the browser's cache memory (See 1 of image below).



Projects stored on board/cache

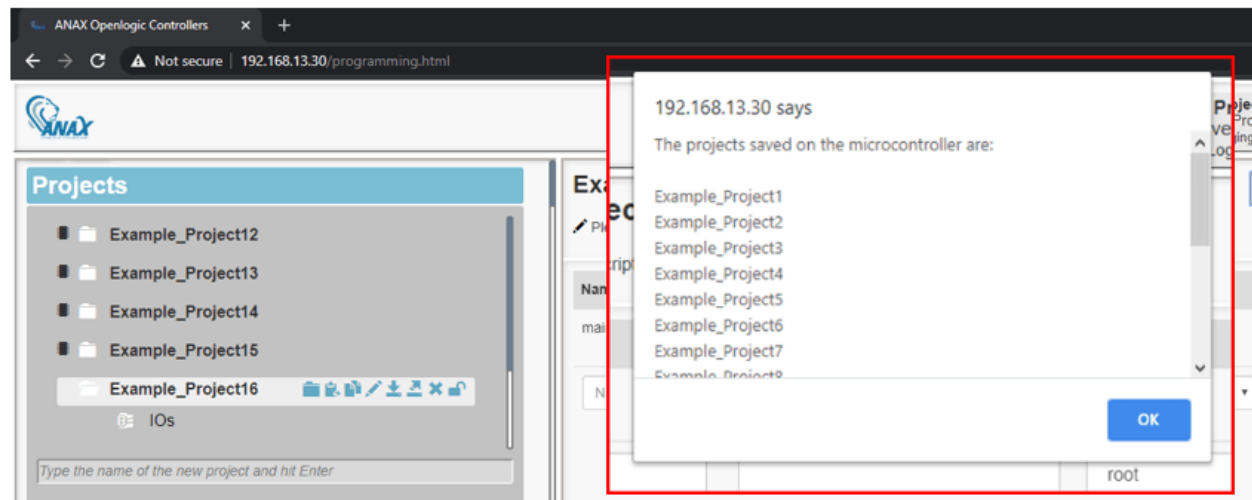
A new project may be created by typing in the input field below the projects list the desired project name. This project is created on the browser's cache and the user may save it later on to the Board.

In case of trying to save more than 15 projects on board, an appropriate message is displayed to the programmer and the action is not allowed (See 1 of image below).



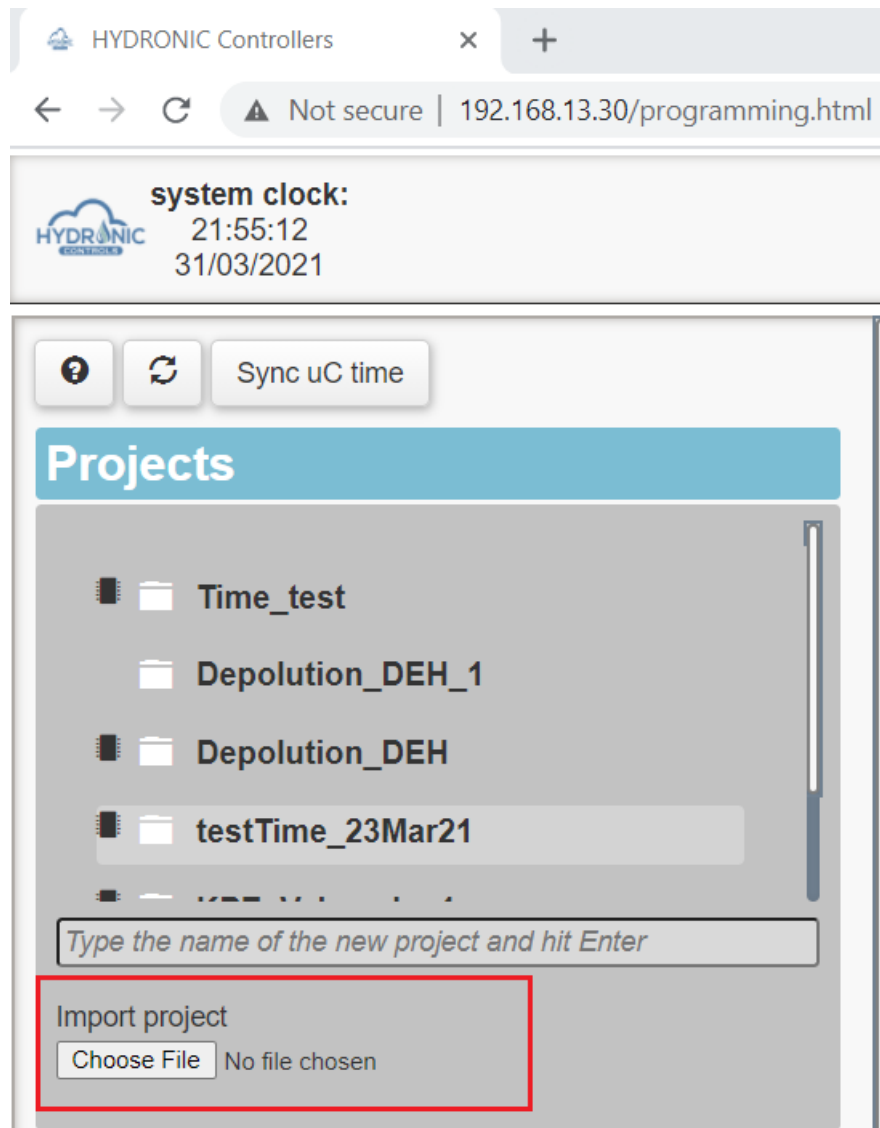
Maximum number of projects on board reached

A list of all the projects saved on board is also displayed (See image below).



List of projects saved on board

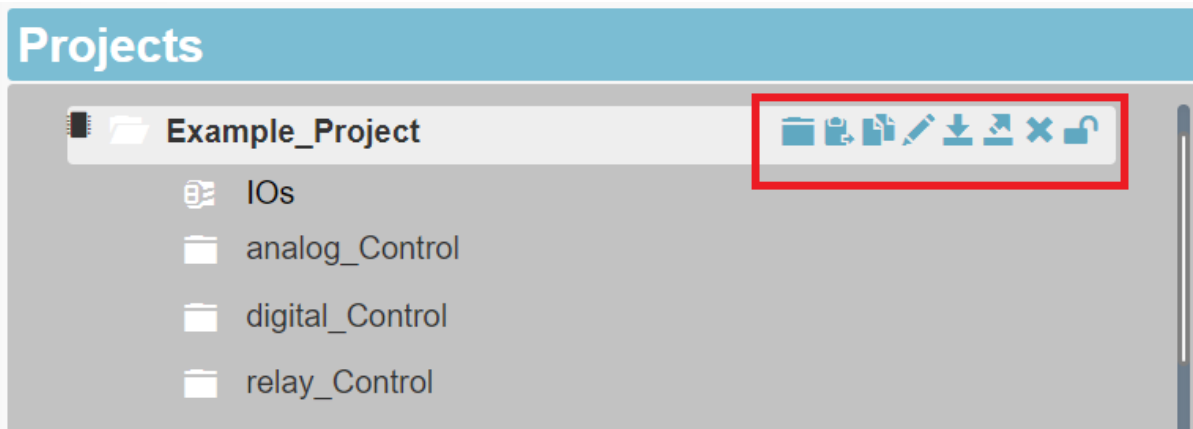
There is also the option to import a project, by clicking the "Choose file" button under the "Import project" label (See image below). An appropriate window will open and ask the user to indicate the *.json project to be imported.



Import project functionality

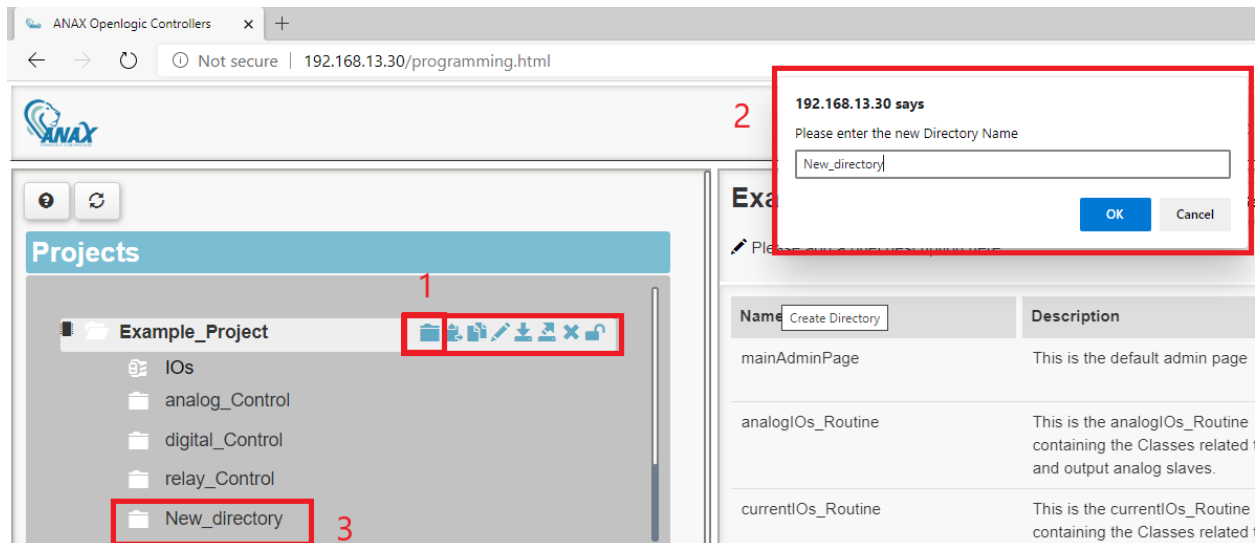
4.8.3 Project Controls

Regarding the project options, the programmer can choose one of the following (See image below)



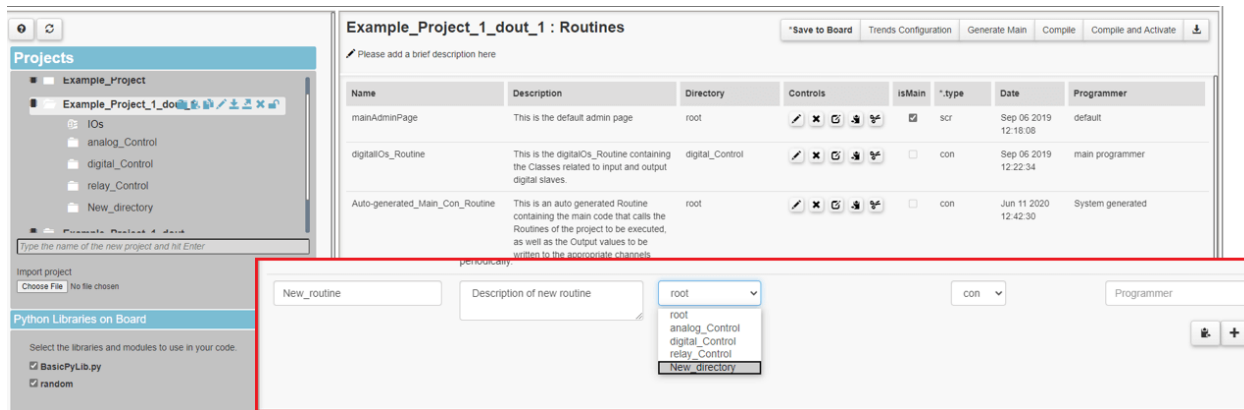
Project options of Hydronic controller

By clicking the first icon the programmer can create a new directory in the currently selected project (See 1 of image below). A new window opens for the programmer to input the name of the directory (See 2 of image below). Upon clicking OK, the new directory gets created (See 3 of image below)



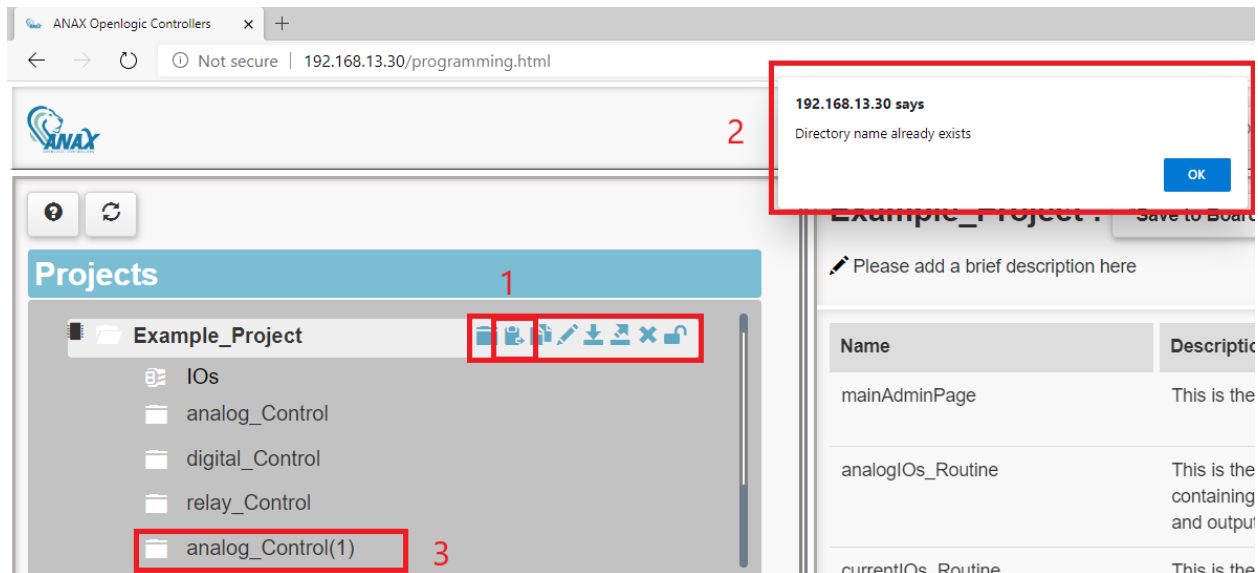
Create directory functionality of Hydronic controller

Now you can add the routines you create, by selecting the created directory in the dropdown menu (See Image below)



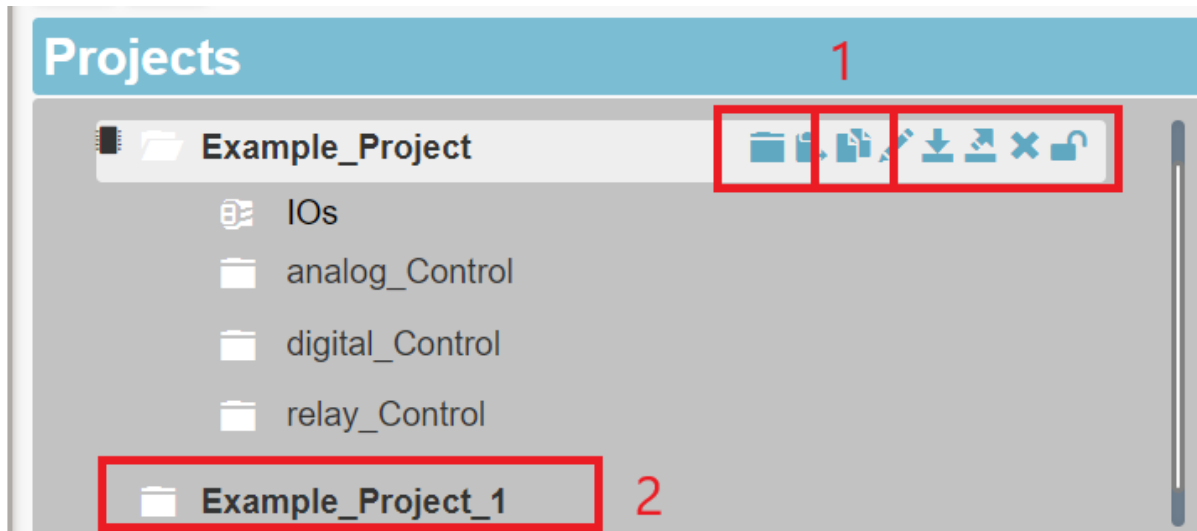
Add routine to directory functionality of Hydronic controller

The second icon, is used to paste a previously copied directory in the currently selected project (See 1 of image below). Upon clicking this icon, if the selected project does not already contain a directory with the same name, the copied directory is pasted, else a new window opens to alert the programmer (See 2 of image below) and the copied directory gets pasted with the characters '(1)' as suffix (See 3 of image below).



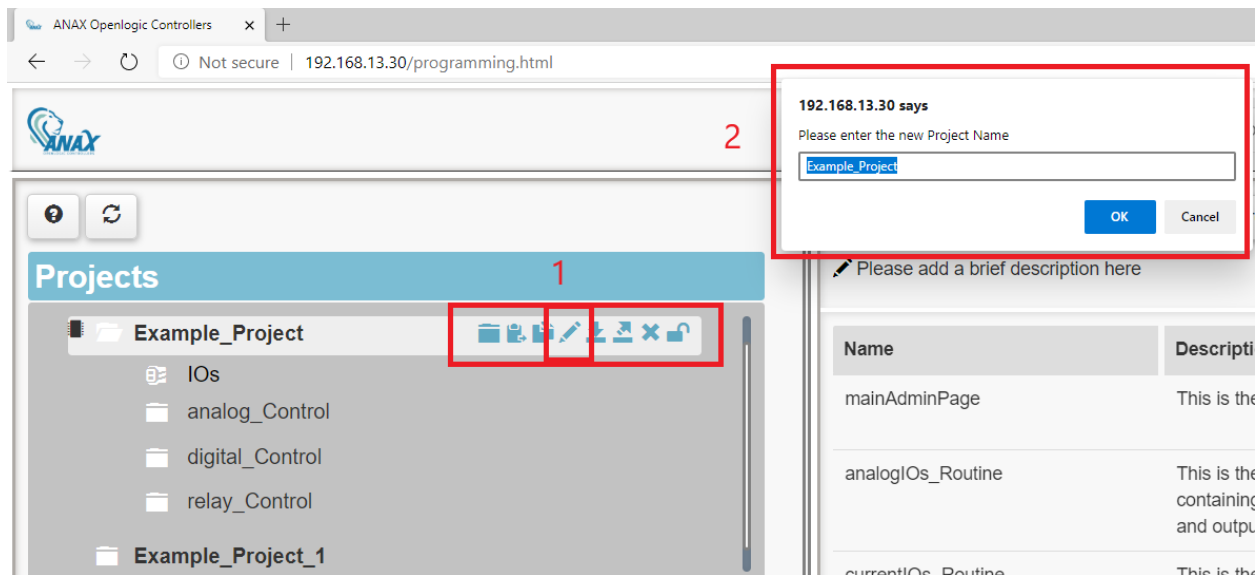
Paste directory functionality of Hydronic controller

The programmer can use the third icon to duplicate the currently selected project with all its directories (See 1 of image below). The characters '_1' act as suffix for the new project (See 2 of image below).



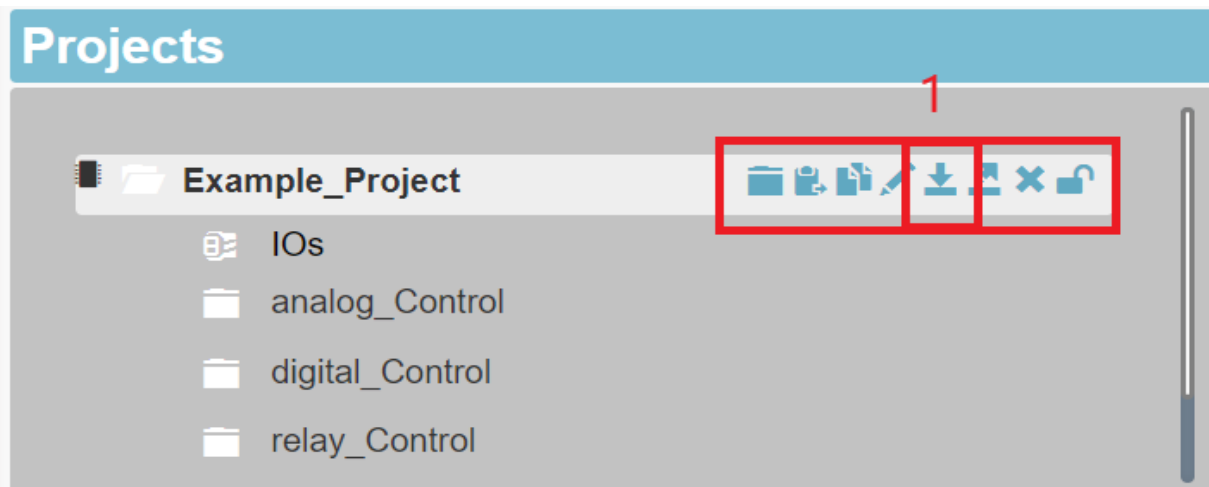
Duplicate project functionality of Hydronic controller

By clicking the fourth icon with the pencil symbol, the programmer can rename a project (See 1 of image below). Upon clicking this icon, a new window opens to input the new name of the project (See 2 of image below).



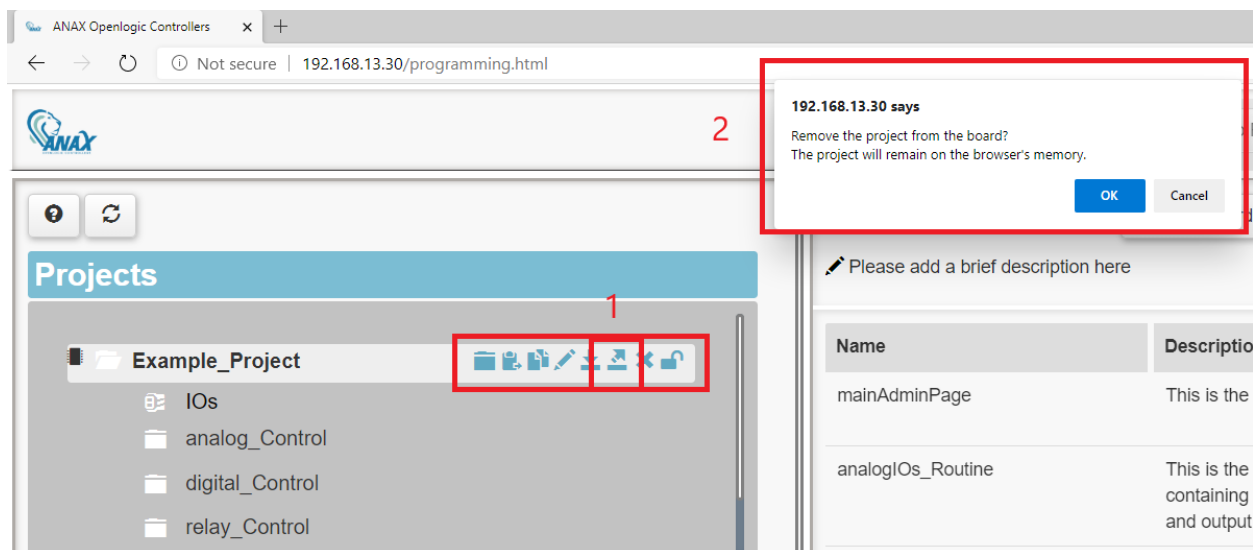
Project rename functionality of Hydronic controller

The fifth icon is used to download the currently selected project locally in *.json format (See 1 of image below). The project is encoded so that the code behind is not readable and the downloaded file may be shared to other CON64A devices for Import.



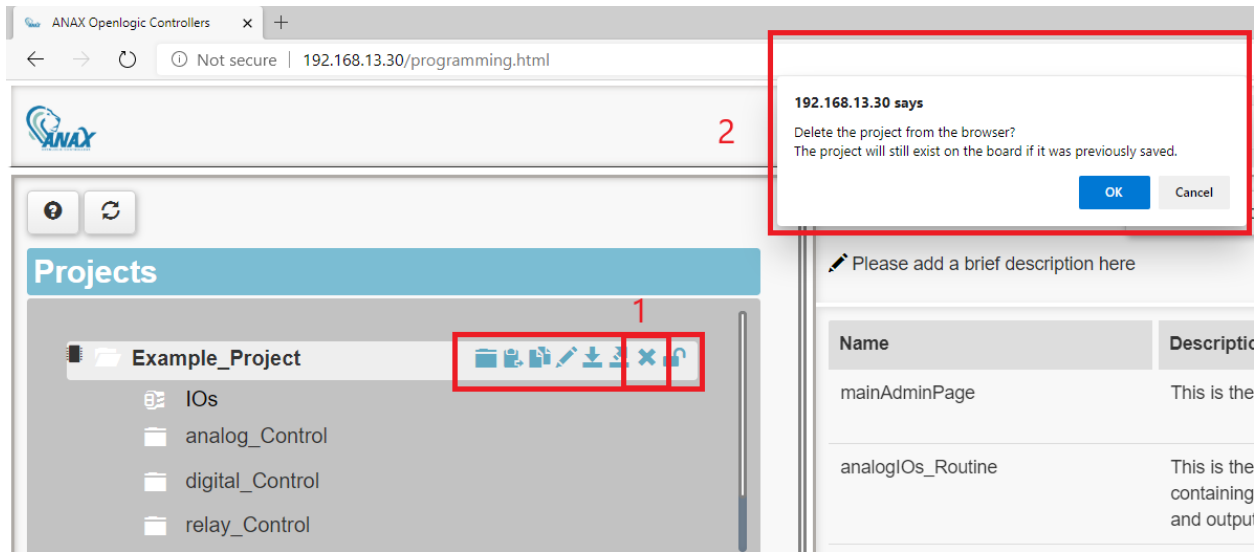
Download project functionality of Hydronic controller

By clicking the sixth icon the programmer can remove the currently selected project from the uC (See 1 of image below). Upon clicking this icon, a new window opens to inform the user that the project will remain on browser's cache memory (See 2 of image below).



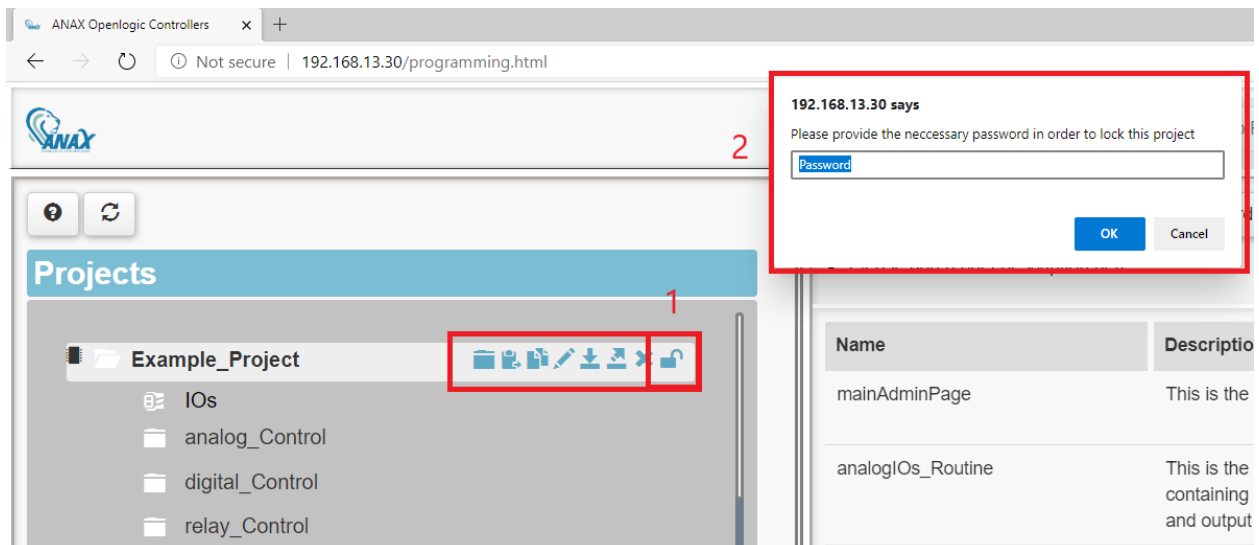
Remove project from board functionality of Hydronic controller

The programmer can use the seventh icon to remove the currently selected project from the browser's cache memory (See 1 of image below). Upon clicking this icon, a new window opens to inform the user that the project will remain on the uC (See 2 of image below).



Remove project from browser functionality of Hydronic controller

The last icon is used to lock the currently selected project (See 1 of image below). Upon clicking this icon, a new window opens so the user can provide a password in order to lock this project (See 2 of image below). This functionality ensures that a different programmer who doesn't know the password will not be able to edit this project.



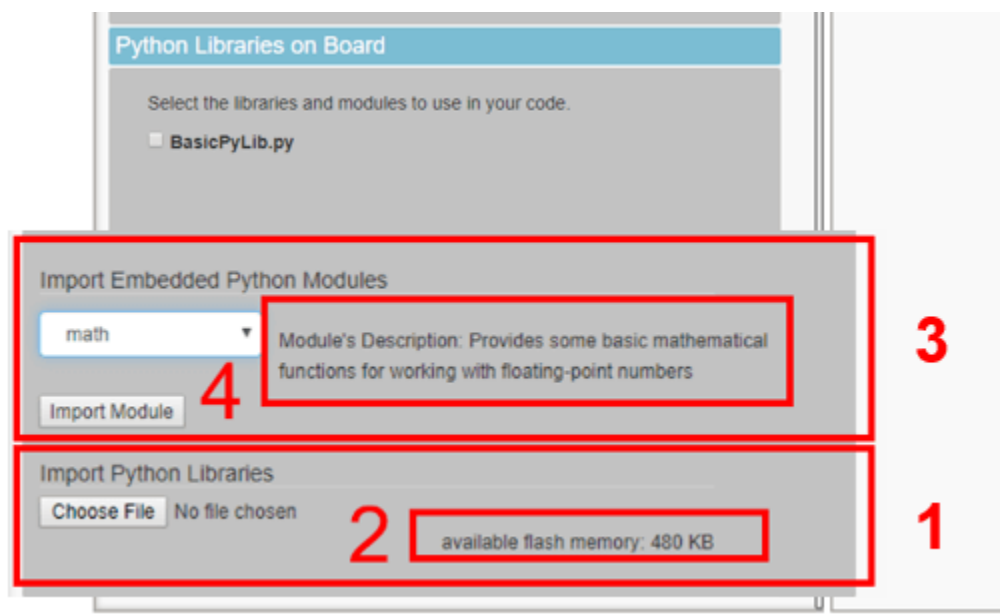
Lock project functionality of Hydronic controller

4.8.4 Python Libraries

The python libraries may be used to allow the programmer to use predefined functions inside his code. They are divided in the ones embedded in the microcontroller and the Imported ones.

The programmer can import and delete python libraries from the related section on the bottom left corner of the main programming window (See 1 of image below). The available flash memory that can be occupied by python libraries is allocated to 480KB (See 2 of image below). Upon importing a new library, the programmer is informed about the memory that will be occupied by this library and total available flash memory bytes are updated. Each library will occupy at least 32KB on uC.

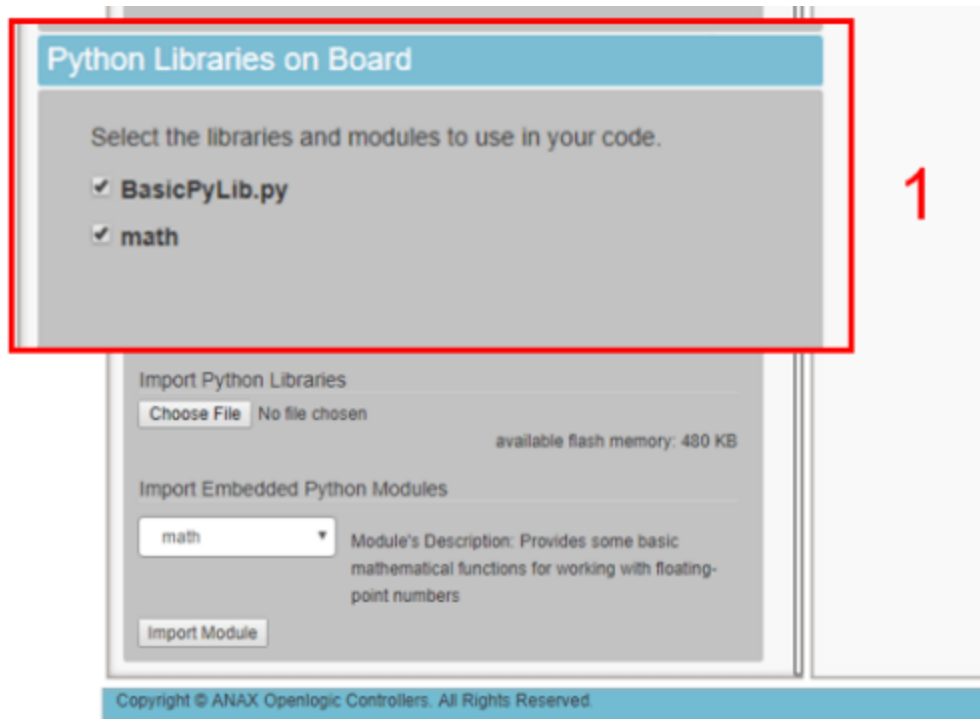
The programmer can also import and delete embedded python modules from the related section on the bottom left corner of the main programming window (See 3 of image below). A variety of python modules are supported. More details on the supported modules and how to use them are described in this document's section '[Python modules](#)'. Upon selecting a python module from the list, a sort description of the module is displayed. (See 4 of image below)



Sections of importing python libraries and modules

Both the imported python libraries and modules are added to the list of python libraries on board. An initially selected imported library or uptyhon module remains selected until the programmer changes its state.

As the list of python libraries is depending only on the uC the checked options have to be adjusted when shifting among multiple projects. This means that the selected libraries are not refreshing according to the selected project. In other words, each library must be selected or deselected from this list before the programmer performs the generation of the main routine, in order to include it or not in the code. (See 1 of image below).



Python Libraries on Board section

4.9 Project IOs Definition Section

In the IOs Definition section, the user can define the project specific behavior of each IO device and shall not be confused with the IO Configuration where the user declares the physical setup of the IO devices. The IOs Definition will be used from each project during Activation state in order to verify that the defined IOs in the project match the declared physical IOs.

The IOs Definition section, is accessed by clicking the IOs label under a project and displays a view divided into four columns with the labels 'Name', 'UniqueID', 'Description' and 'Controls'. A new IO can be added to the definitions list by filling the related fields and clicking the plus button (See 1 of image below). Explanation hover texts are displayed for all fields.

The list of IOs can be deleted by the 'trash' button on the bottom right corner of the section (See 2 of image below). The programmer can edit the 'UniqueID' and 'Description' fields. Upon making any changes on these fields the 'save' button on the bottom right corner is enabled and the programmer can click it in order to save the changes in the browser's cache (See 3 of image below).

On the top right corner of the IOs Definition section three buttons are available. (See 4 of image below).

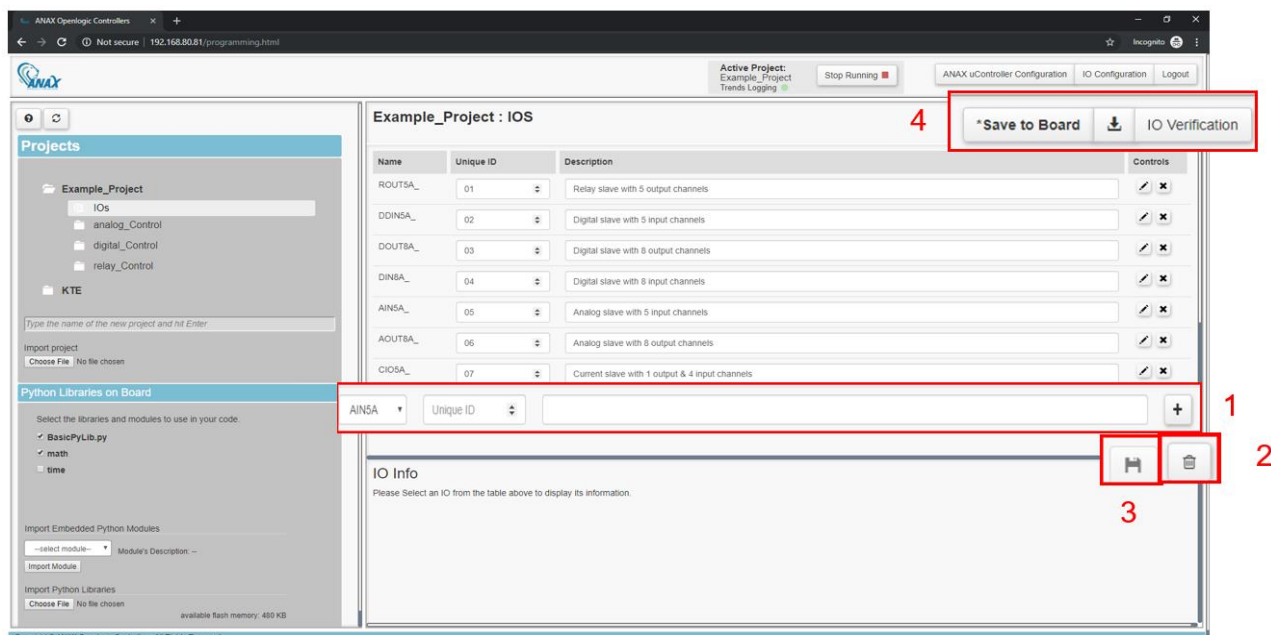
The 'Save to Board' is enabled when unsaved changes exist for the related project on the browser's cache. This button saves to the board, all existing changes of the related project.

The 'IO Verification' button is used to verify that the IOs added in the IOs Definition section match the IOs added in the IO Configuration section by searching their Unique IDs, so that the programmer can proceed later on with the activation of his project.

IO Verification is valid only if the IO can be found in the IO Configuration list which is successfully checked by the system. Project IOs may be a subset of the IO Configuration.

If this procedure finishes successfully then the programmer is prompted to download the list of the configured IOs. If not, then a message appears in the "IO Info" section and he is suggested to navigate to IO Configuration section to correct the declared IOs.

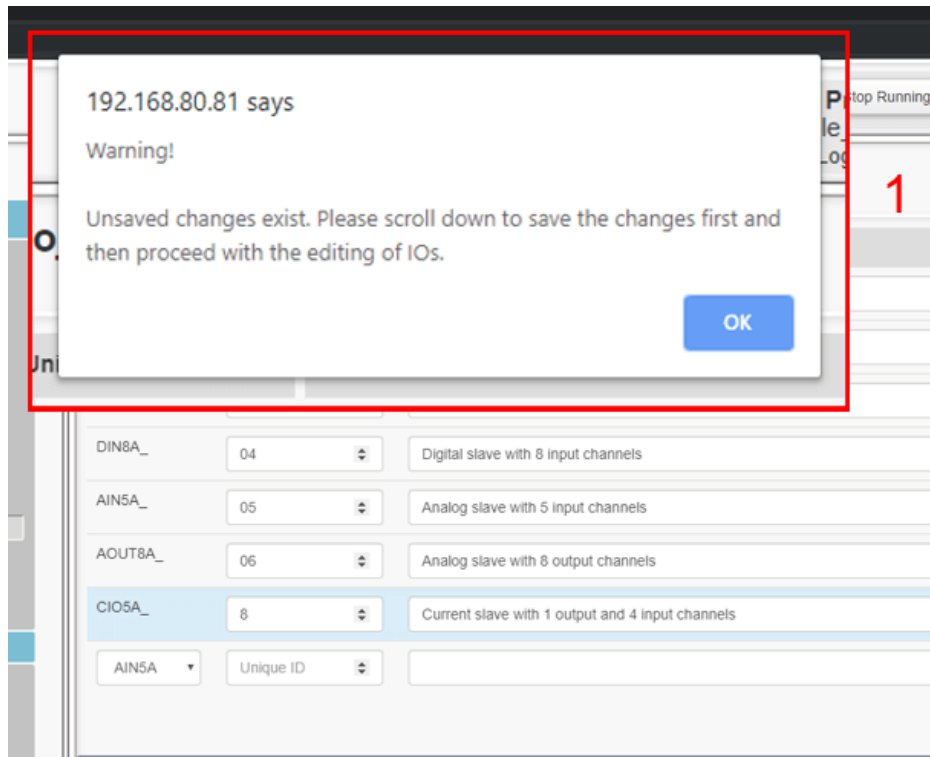
By clicking the button with the download icon, the programmer can download the IOs Definition of the specific project in *.csv format and use this file to set up IO Configuration accordingly. This feature though, should be used after the configuration of the channels of the IOs have been made by the programmer.



IOs Definition section of Hydronic controller

The programmer can edit an IO by clicking the 'Configure IO' button which is the first one of the Control buttons for each IO.

If unsaved changes exist on IOs Definition the programmer is not able to edit an IO. An alert window is displayed and the programmer is guided in order to first save the changes (See 1 of image below).



Unsaved changes on IOs Definition section

If no unsaved changes exist on IOs Definition, the edit window for the specific IO opens.

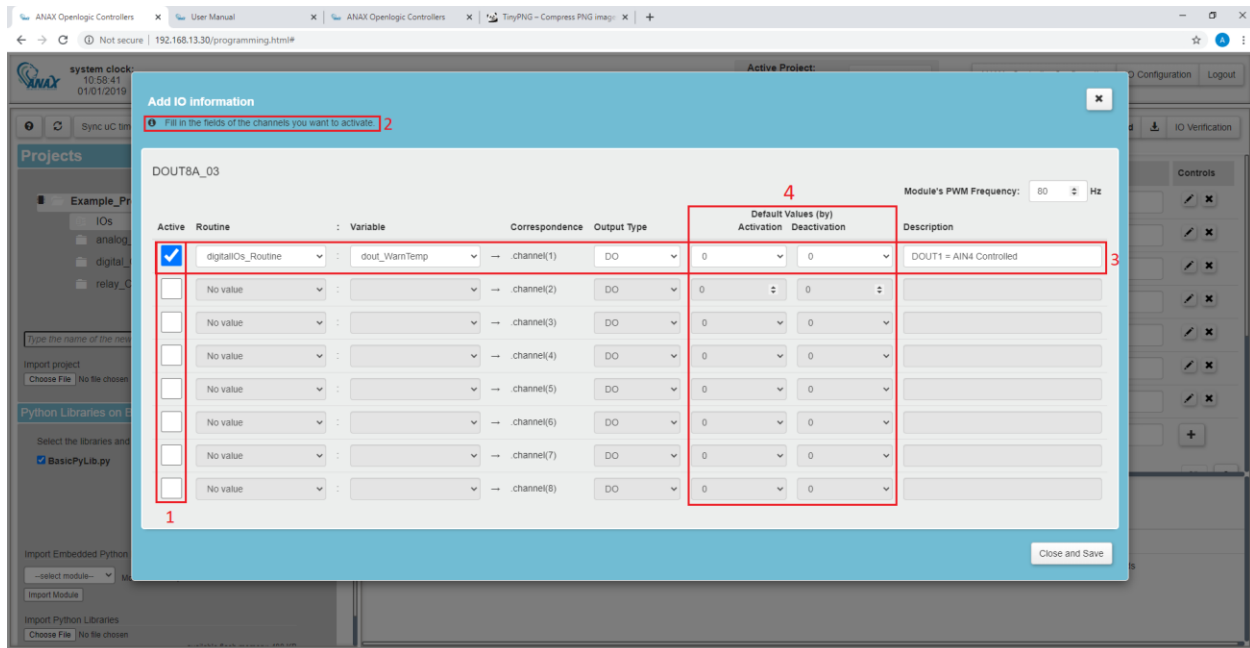
A checkbox for each channel to the edit window of an IO is used to activate or deactivate a channel. All channels are initially inactive and cannot be edited (See 1 of image below).

A user prompt with usage instructions for each IO is displayed on the top of the edit window. (See 2 of image below).

Upon activating a channel, the programmer can edit all fields related to the specific channel and save the changes in order to take effect (See 3 of image below).

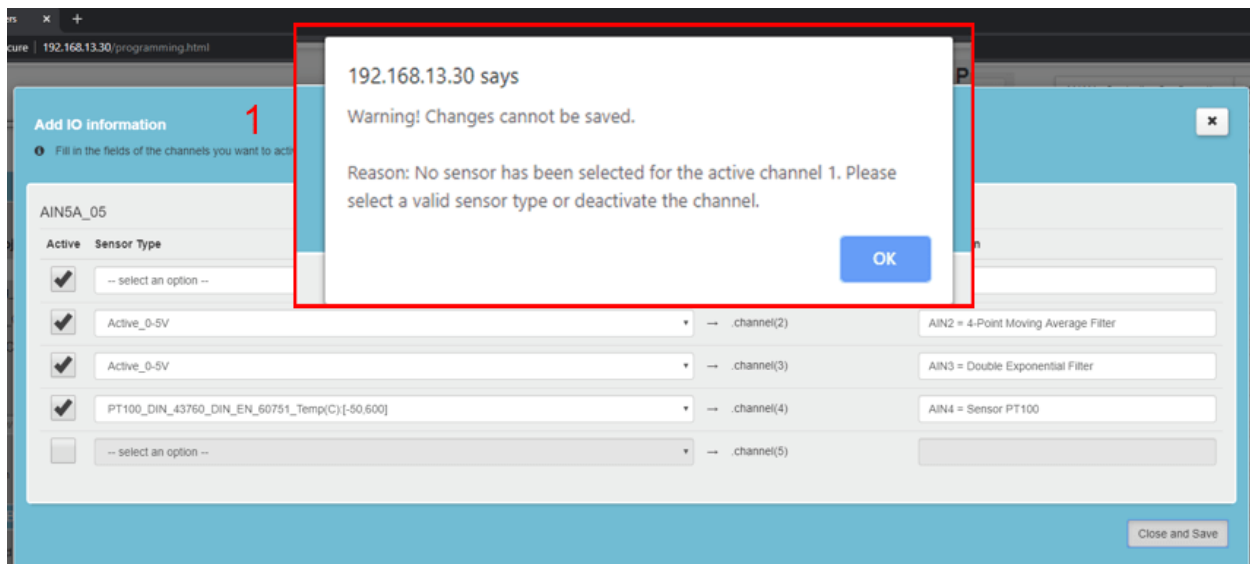
For all the Output IO types two fields are used for the Activation and Deactivation Default Values. The Default Activation values are used each time the project is activated and the Default Deactivation values are set each time the project is deactivated, having in that way a safety values functionality, so that the project is never found in an unwanted state. The initial value for both fields is set to zero. (See 4 of image below)

In case a given value does not comply to the above restrictions, a message is displayed to the programmer upon 'Close and Save'. The programmer has to correct the fields in order to successfully save the changes.



Preview of the IO edit section upon activation of a channel

For the AIN5A IO especially, a valid sensor type must be selected from the sensors' list in order for a channel to be activated (See 1 of image below).

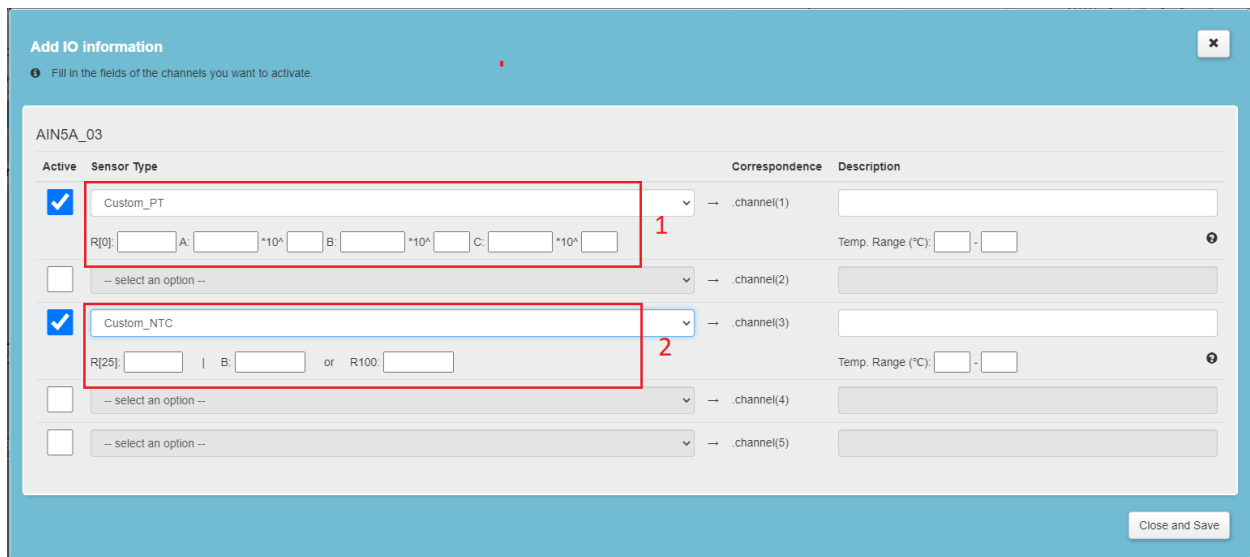


AIN5A channel activation

In the same list four custom sensor types are available to choose from in case the given sensor options do not meet the desired criteria.

Custom PT: The PT equation used to calculate the PT sensor values is $R = \text{Type}(1 + A \cdot t + B \cdot t^2 + C \cdot (t-100) \cdot t^3)$. The programmer is required to fill the fields referred to the characteristic values of the sensor (See 1 of Image below), along with the measured Temperature range.

Custom NTC: The NTC equation used to calculate the NTC sensor values is $R(T) = R[25] \cdot e^{B \cdot (1/T - 1/T[25])}$. Again, the programmer must fill the necessary fields to customize the NTC sensor (See 2 of Image below), along with the measured Temperature range.



The screenshot shows the 'Add IO information' dialog box with the title bar 'AIN5A_03'. Below the title bar is a header 'Fill in the fields of the channels you want to activate.' The main area contains a table with columns 'Active', 'Sensor Type', 'Correspondence', and 'Description'. The table has five rows. The first row is for 'Custom_PT' (Active: checked, Sensor Type: Custom_PT, Correspondence: channel(1), Description: Temp. Range (°C): [] - []). A red box labeled '1' highlights the input fields for R[0], A, B, and C. The second row is for 'Custom_NTC' (Active: checked, Sensor Type: Custom_NTC, Correspondence: channel(3), Description: Temp. Range (°C): [] - []). A red box labeled '2' highlights the input fields for R[25], B, and R100. The third, fourth, and fifth rows are for 'select an option' (Active: unchecked, Sensor Type: -- select an option --, Correspondence: channel(2), channel(4), and channel(5) respectively, Description: empty).

Custom PT and NTC sensors

Custom NI: The Ni equation used to calculate the Ni sensor values is $R = \text{Type}(1 + A \cdot t + B \cdot t^2 + C \cdot t^3 + D \cdot t^4)$. The programmer is required to fill the fields of the parameters of the sensor (See 1 of Image below).

Custom Linear Sensor: In this case the programmer is asked to define 3 different pairs of values (X, Y) in order a linear approach to be performed. (See 2 of image below).

Add IO Information

Fill in the fields of the channels you want to activate.

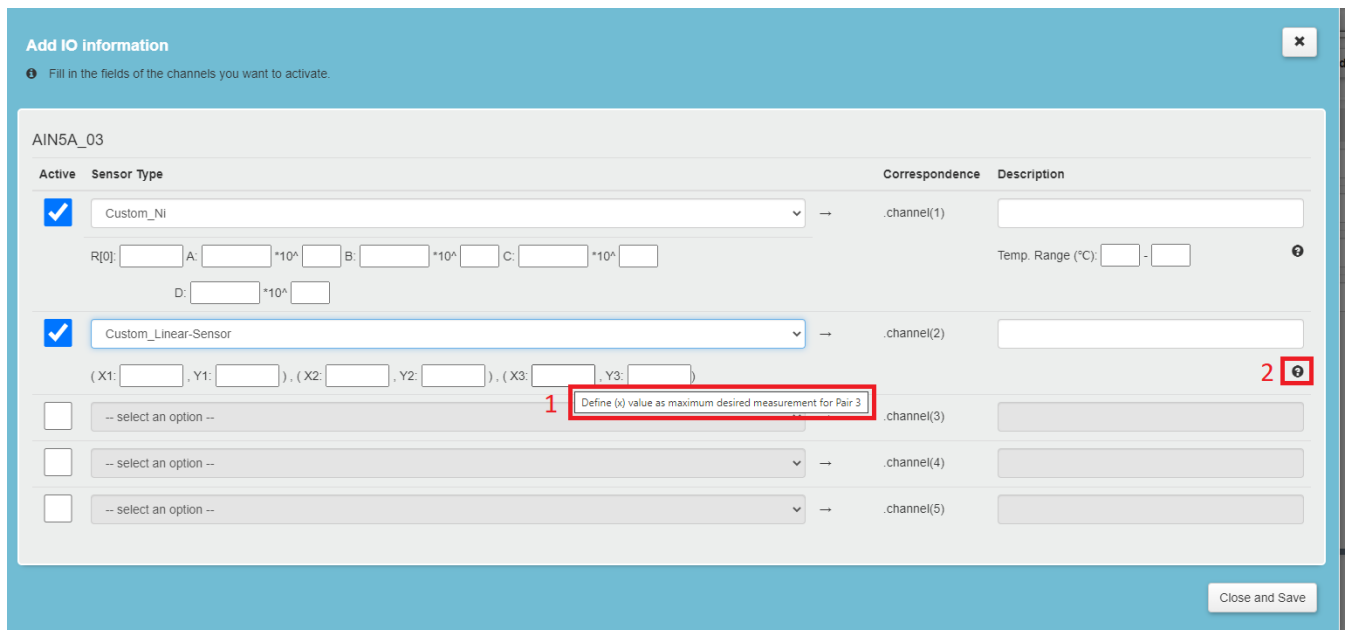
AIN5A_03

Active	Sensor Type	Correspondence	Description
<input checked="" type="checkbox"/>	<div>Custom_Ni</div> <div> R[0]: <input type="text"/> A: <input type="text"/> *10^A B: <input type="text"/> *10^A C: <input type="text"/> *10^A D: <input type="text"/> *10^A </div>	→ channel(1)	<input type="text"/> Temp. Range (°C): <input type="text"/> - <input type="text"/>
<input checked="" type="checkbox"/>	<div>Custom_Linear-Sensor</div> <div> (X1: <input type="text"/> , Y1: <input type="text"/>) , (X2: <input type="text"/> , Y2: <input type="text"/>) , (X3: <input type="text"/> , Y3: <input type="text"/>) </div>	→ channel(2)	<input type="text"/>
<input type="checkbox"/>	-- select an option --	→ channel(3)	<input type="text"/>
<input type="checkbox"/>	-- select an option --	→ channel(4)	<input type="text"/>
<input type="checkbox"/>	-- select an option --	→ channel(5)	<input type="text"/>

Close and Save

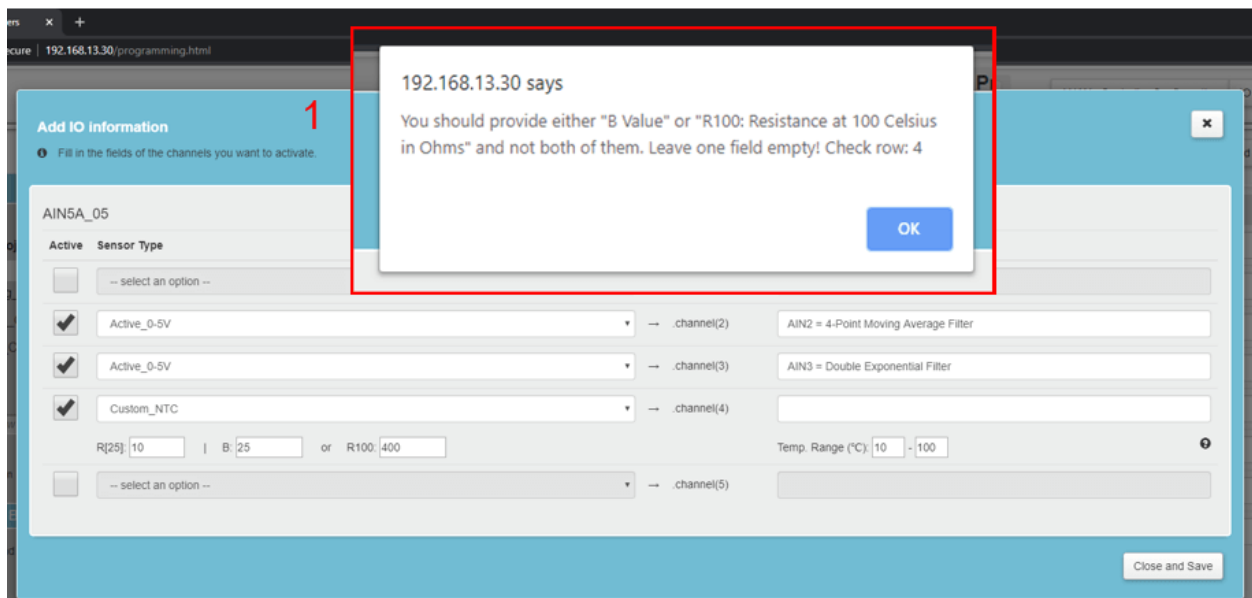
Custom NI and Linear Sensors

More information about the custom sensor can be seen by hovering above the desired field (See 1 of Image below). Additionally, with mouse hovering above the question mark icon (See 2 of Image below) the equation of each custom sensor is displayed.



Custom Sensors Fields Explanation

In case of invalid custom sensor fields, the 'Close and Save' button is disabled and an appropriate message is displayed (See 1 of image below). The programmer has to correct the fields in order to successfully save the changes.



AIN5A custom sensor validation

4.10 Activation/Deactivation IO states

In this section the activation/deactivation Output channel states of the IOs are described.

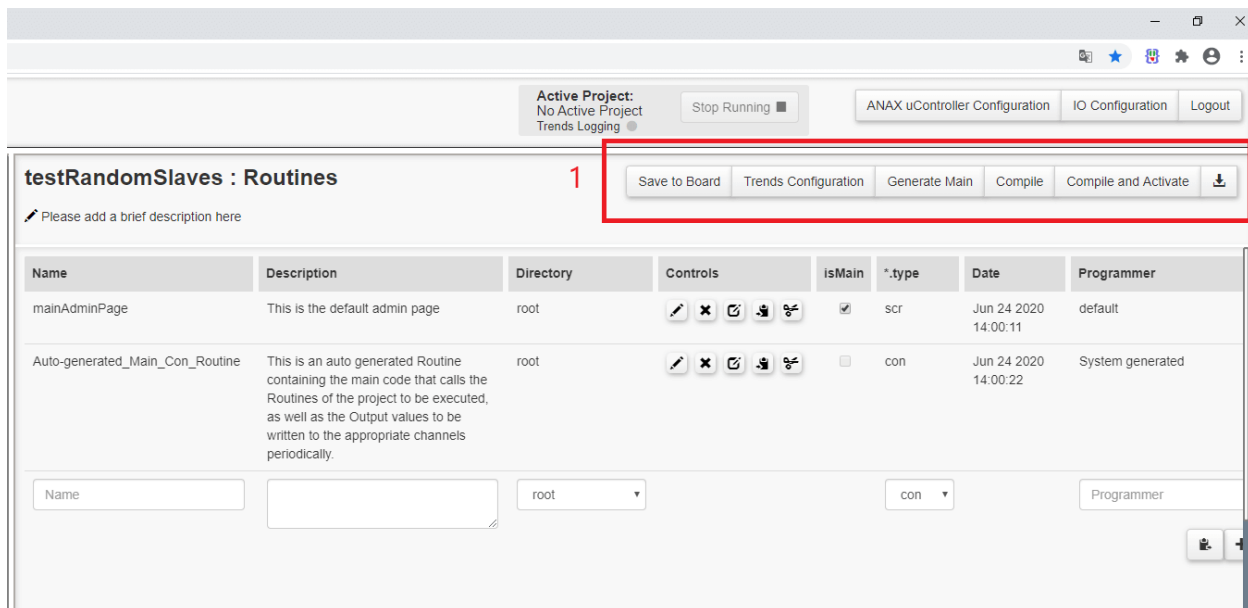
1. Upon the very first power up of the uC, all channel values are set to **Zero values**.
2. Upon the activation of project either as programmer or as admin user, all channel values are set to the **Default Activation** values.
3. Upon deactivation of a project either as programmer or as admin user, all channel values are set to the **Default Deactivate** values.
4. When a project is active and a power down of the uC happens, when power is restored, all channel values are set to **Default Activation** values and the project functionality is resumed.
5. When a project is deactivated from the programmer and the uC is powered down, when power is restored all channel values are set to Zero Values.

When a project is deactivated from the admin and the uC is powered down, when power is restored all channel values are set to **Default Deactivate** values

4.11 Project Configuration section

4.11.1 Action buttons

On top right corner of each project's section the programmer can see 6 buttons which are related to the handling and activation of the python code (See 1 of image below).

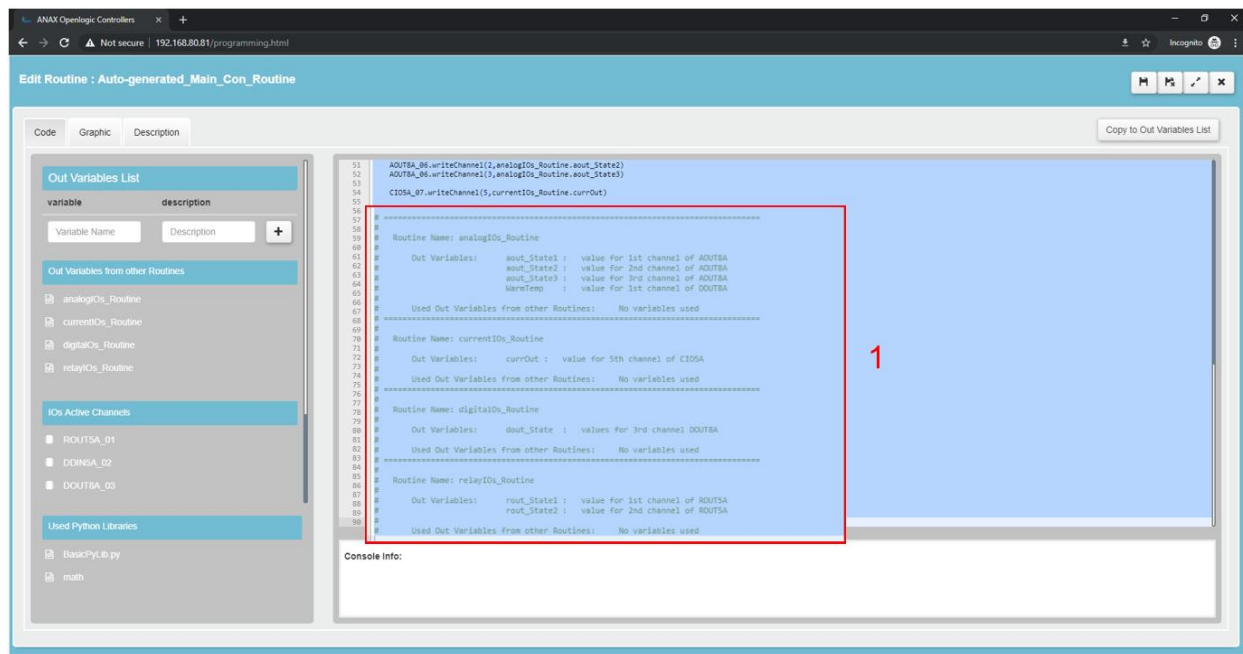


Project Section buttons presentation

The "Save to Board" button is used to save to the board all existing changes of the related project.

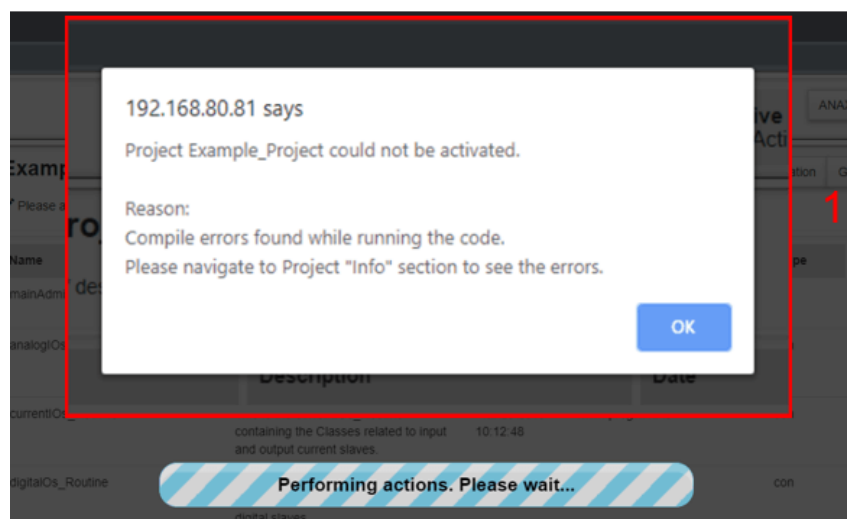
The "Trends Configuration button" opens the modal with the settings related to the project's Trends handling but it's explained later on in detail, in a [section](#) of its own.

The “Generate Main” button generates a main routine for the project which merges all the available routines defined by the programmer in one. This main routine is responsible for running the project code in a loop. If the main routine exists already, a copy of the existing main routine is created as the _OLD version and a new main routine is generated. The generation of the main routine can be performed regardless of the existence of the IO Configuration. Additionally, inside the code of the main routine on the bottom section there are in comments detailed information about the variables used by each routine. (See 1 of image below).



Auto-generated_Main_Con Routine section of out variables

The button ‘Compile’ performs the syntax check for the written python code. The code of all routines along with the main routine are being merged appropriately and the syntax check is done to the merged python code. Any detected syntax errors (See 1 of image below)



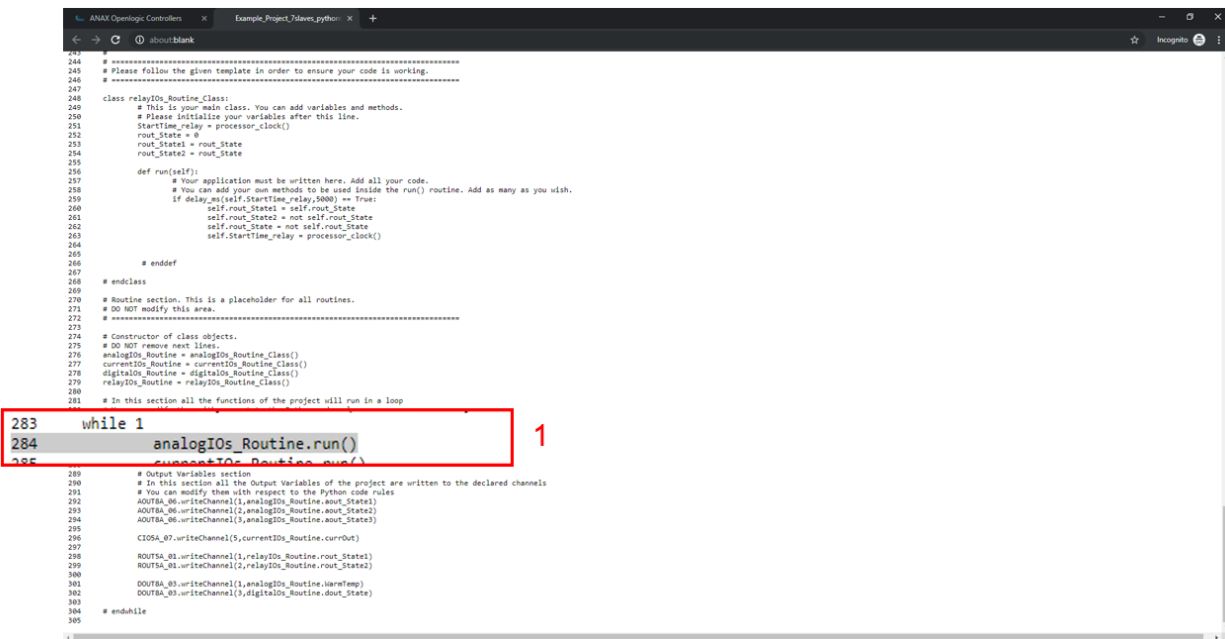
Alert window after syntax check of the python code

are displayed on the Info section of the related project right after the syntax check completes (See 1 of image below).



Console log of the error Information

In case of existing syntax errors, the merged python code will also open in a new tab with the error line marked (See 1 of image below).



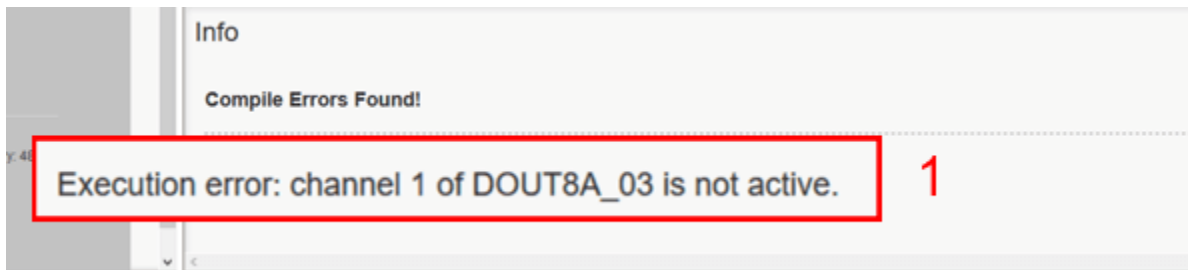
Python code in a new tab with a marked syntax error

When clicking 'Compile' the syntax check is performed regardless of the existence of the IO Configuration. 'Compile' functionality requires no active project, hence, if an active project exists it should be first deactivated by the user.

The fifth button in the row is the 'Compile and Activate' button. By clicking this button all procedures related to the activation of a project are performed in series. These procedures consist of the following in the order being listed: IO Addressing, IO Verification, Compile and Activation of the project. In order to perform 'Compile and Activate', programmer should have

Configure the IOs first in the IO Configuration tab. 'Compile and Activate' functionality requires no active project, hence, if an active project exists it should be first deactivated by the user.

Logical checks have been added during the activation procedure. These checks are related to the use of IOs and their channels inside the python code. The use of any undefined IO or any non-active channel inside the python code is detected and the appropriate error message is displayed (See 1 of image below).



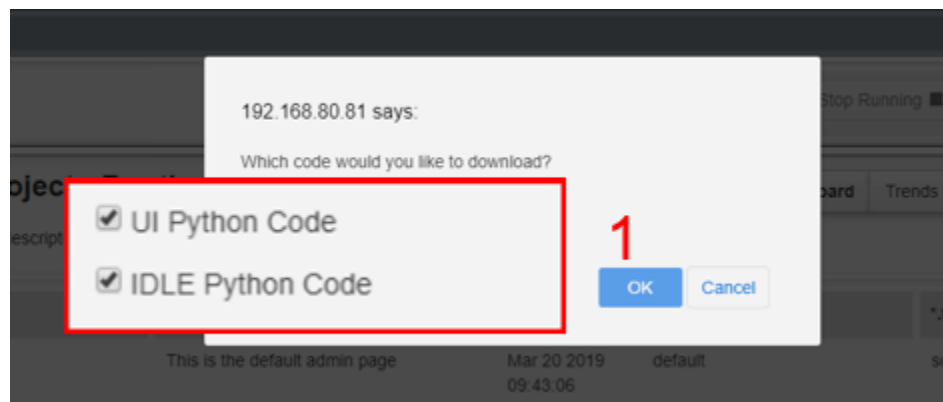
Logical error detected inside python code

When 'Compile and Activate' function is completed, the python code will be running on the Controller, given that no syntax or logical errors were detected during the procedure.

It has to be mentioned here that the memory allocated to compile the project is specific for the ucontroller. It is specified up to 64k and each 10 Lines of Code (LOC) occupy 1k. This means that a code over 640 LOCs cannot be compiled. As this memory is fixed a message is thrown when the user tries to compile the project and the lines detected are above 620 LOCs.

Additionally, in case of such an error the Project Info section is updated accordingly to indicate the Memory error and suggest the user to correct the code to be compiled. This error message indicates: "Warning! The effective python code (omitting any comments) exceeds the 620 limit of Lines Of Code. Please use Libraries and functions to make the code shorter and efficient, otherwise your code may not be compiled."

A download button has been added in the last position of the row. The download can be performed regardless of the activation state of the project. This button offers the programmer two download options (See 1 of image below). He can opt for both of them or separately.



Alert window for downloading python code

The first option 'UI Python code' downloads a python file that contains the merged python code as seen in the UI. In order to download the python code, it is required that the Auto-generated_Main_Con_Routine exists.

The second option 'IDLE Python code' supports the functionality to download the written python code for debugging in the IDLE integrated development environment for python. This option downloads a zip folder that contains two files. The first one is the merged python code modified appropriately for IDLE environment and the second is a python file with the name 'IDLE.py' which contains dummy implementations of basic functions unsupported by IDLE. These files must be saved in a folder where the IDLE editor has access.

The necessary software tools for the execution of the IDLE Python Code are:

1. Python 3.6 or newer version
2. Software to handle zip files

In order to import the downloaded code to the IDLE environment, the following steps shall be executed:

1. Select from Windows **Start -> Python -> IDLE**. The IDLE Python Shell window will be opened
2. Check the folders that the Python IDLE has access. They are listed with the execution of the following commands in the IDLE Python Shell window:
 - a. `>>>import sys`
 - b. `>>>sys.path`
3. Unzip the downloaded files into a desired folder. You shall have two files.
4. The **Example_Project_IDLE.py**
5. The **IDLE.py**
6. If the files downloaded are stored in a non-existing path, add this **<Your Path>** with the following commands:
 - a. `>>>import sys`
 - b. `>>>sys.path.append("<Your Path>")`
7. Select **File -> Open** and Choose the downloaded **Example_Project_IDLE.py** file. It will open in a new editor window.
8. Select **Run -> Run Module** or press **F5**. Then, the **Example_Project_IDLE.py** is executed.

After successful execution the following message will appear in the IDLE Python Shell window.
===== RESTART: < Your Path >\Example_Project_IDLE.py =====

4.11.2 Routine Entries

A routine is easily created by completing the related fields in the Routines section and clicking the + button. Then a new entry in the list will appear.

A routine may either be a CON or an SCR type.

SCR routines are supporting only the design of graphics. An SCR routine may be marked as “isMain” in order to be the first one to open when loading the Run page of an Active Project.

CON routines are the main coding routines of the system. The program may use them to create code define Output variables and add graphic pages related to them too.

4.11.3 Routine Control Buttons

There are five control buttons for each routine of a project.

Example_Project : Routines							
Please add a brief description here				*Save to Board	Trends Configuration	Generate Main	Compile
				Compile and Activate			
Name	Description	Directory	Controls	isMain	*.type	Date	Programmer
mainAdminPage	This is the default admin page	root	1 2 3 4 5	<input checked="" type="checkbox"/>	scr	Sep 06 2019 12:18:08	default
analogIOs_Routine	This is the analogIOs_Routine containing the Classes related to input and output analog slaves.	analog_Control		<input type="checkbox"/>	con	Sep 06 2019 12:21:08	main programmer
currentIOs_Routine	This is the currentIOs_Routine containing the Classes related to input and output current slaves.	analog_Control		<input type="checkbox"/>	con	Sep 06 2019 12:21:57	main programmer
digitalIOs_Routine	This is the digitalIOs_Routine containing the Classes related to input and output digital slaves.	digital_Control		<input type="checkbox"/>	con	Sep 06 2019 12:22:34	main programmer
relayIOs_Routine	This is the relayIOs_Routine containing the Classes related to relay slaves.	relay_Control		<input type="checkbox"/>	con	Sep 06 2019 12:23:08	main programmer
Auto-generated_Main_Con_Routine	This is an auto generated Routine containing the main code that calls the Routines of the project to be executed, as well as the Output values to be written to the appropriate channels periodically	root		<input type="checkbox"/>	con	Sep 06 2019 12:35:42	System generated

Routines Control Buttons

The first button (1) edits the header of the routine. When the programmer clicks this button a pop-up modal opens with the available fields of the routine to edit.

The second button (2) deletes the routine.

By clicking the third icon (3), the programmer is able to edit the code of the routine, as it is described in the paragraph [‘Routines Editor’](#).

With the fourth icon (4), the user creates a copy, in the clipboard, of the routine which he is able to paste with the ‘paste button’ located in the Routine entries view on the bottom.

Finally, the fifth button (5) cuts the routine from the project, moving it to the clipboard. Then it is available for paste.

4.11.4 CON Routines editor

Each CON Routine's editor window consists of a left-side panel which includes useful options for the programmer and a right-side panel which includes the editor of python code.

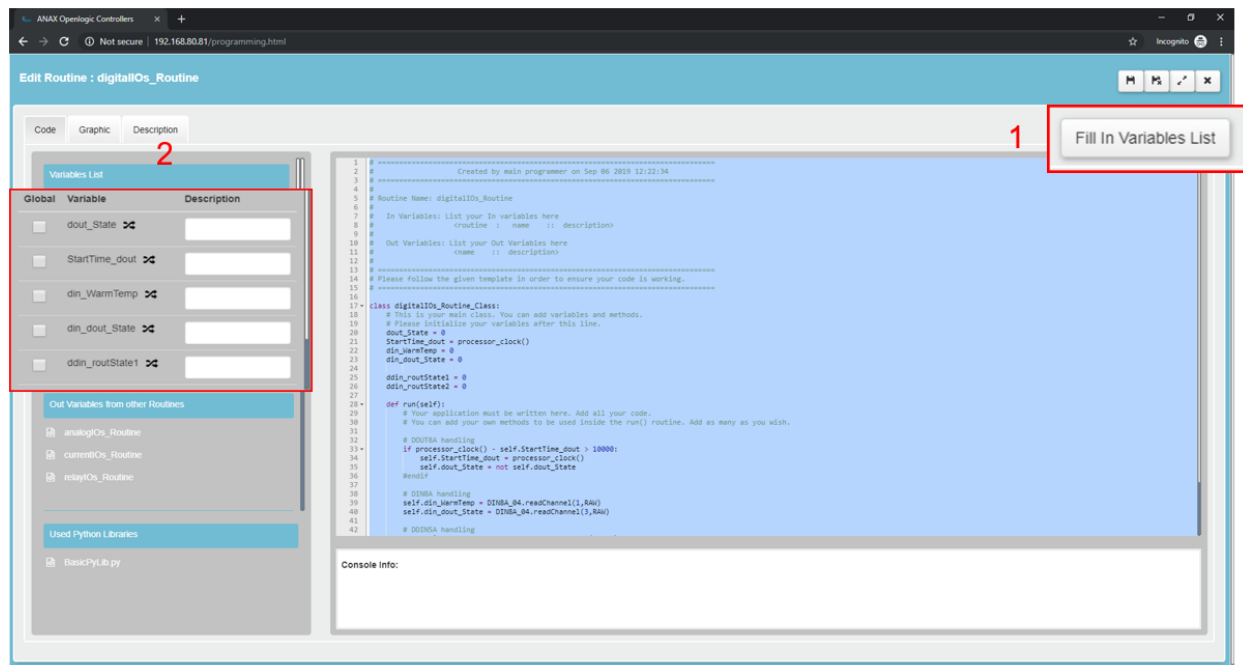
The left-side panel includes the following sections: the 'Variables List', the 'Out Variables from other Routines', the 'IOs Active Channels' and the 'Used Python Libraries' section.

By clicking on a routine name from the 'Out Variables from other Routines' section, the programmer can see and insert in his code the out variables of the specific routine he clicked.

By clicking on an IO from the 'IOs Active Channels' section, the programmer can see the active channels of the specific IO and insert the readChannel function call inside his code.

If the user has imported python libraries to his code, he can see the library code by double clicking on the name of the library as it's been displayed on the 'Used Python Libraries' section.

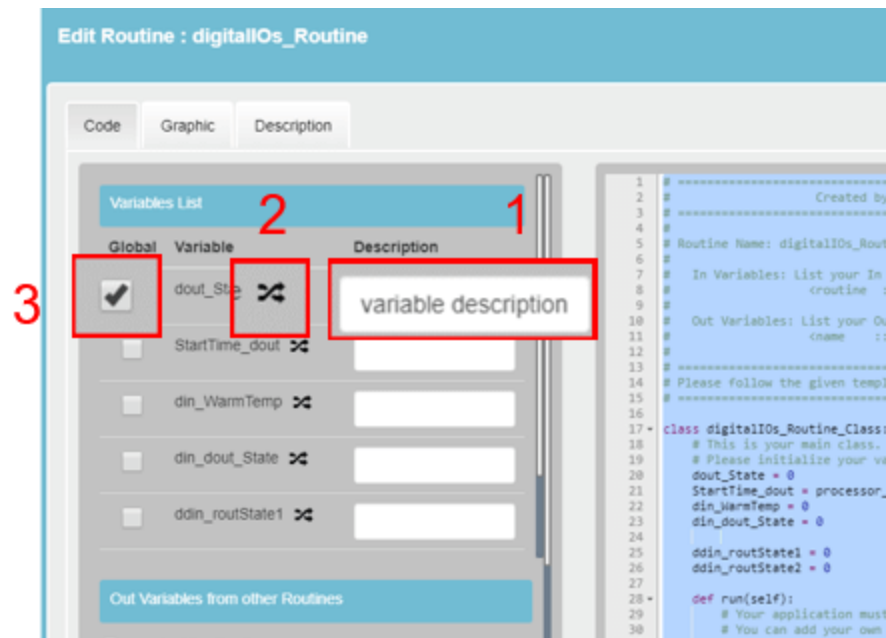
On the top right corner of the window exists the 'Fill In Variables List' button. By clicking on 'Fill In Variables List button' (See 1 of image below), all the initialized variables inside the routine's python code are inserted to the 'Variables List' section (See 2 of image below).



Routines' editor

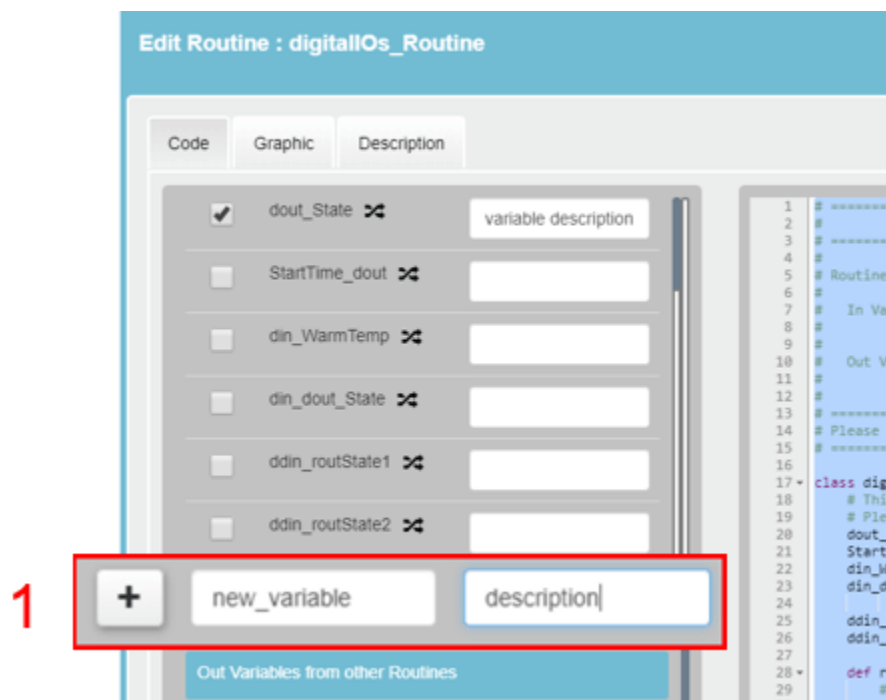
The programmer can edit the description of a variable (See 1 of image below), or insert it inside the code with the appropriate 'self.' prefix by clicking the double arrow button next to the variable's name (See 2 of image below).

Additionally, the programmer can select a variable as global by checking the related checkbox on the left side of the listed variable (See 3 of image below).



Variables List section

The 'Variable Name' and 'Description' fields on the bottom of the section can be used in order for the programmer to insert a variable manually in the List. By clicking the 'plus' button a manually declared variable is inserted in the List and can be declared as global or be edited. (See 1 of image below). In case of an invalid name, the variable is not inserted and the programmer is informed with the appropriate message.



Variables List section

Upon clicking 'Fill In Variables List' button a previously existing list is cleared along with all the declared variables. The list is recreated with the default settings as already described.

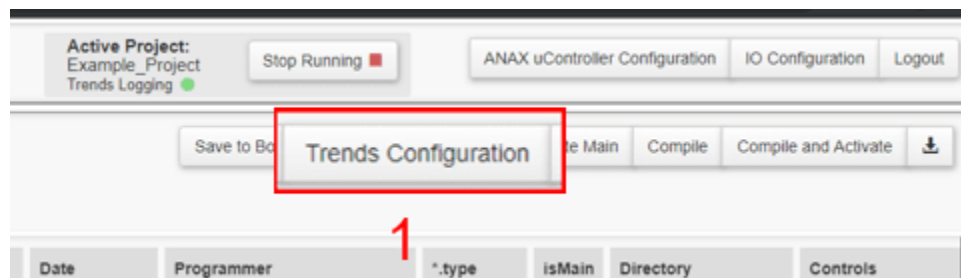
Any changes made, must be saved in the Routine in order to take effect.

Three Tabs are available in the CON Routine's editor window. These give access to the Code section described already, the Graphic section which will be described in the [SCR and CON Routines Graphic Pages](#) section and the Description editor where the programmer may define more details for a routine's description.

4.11.5 Trends Configuration

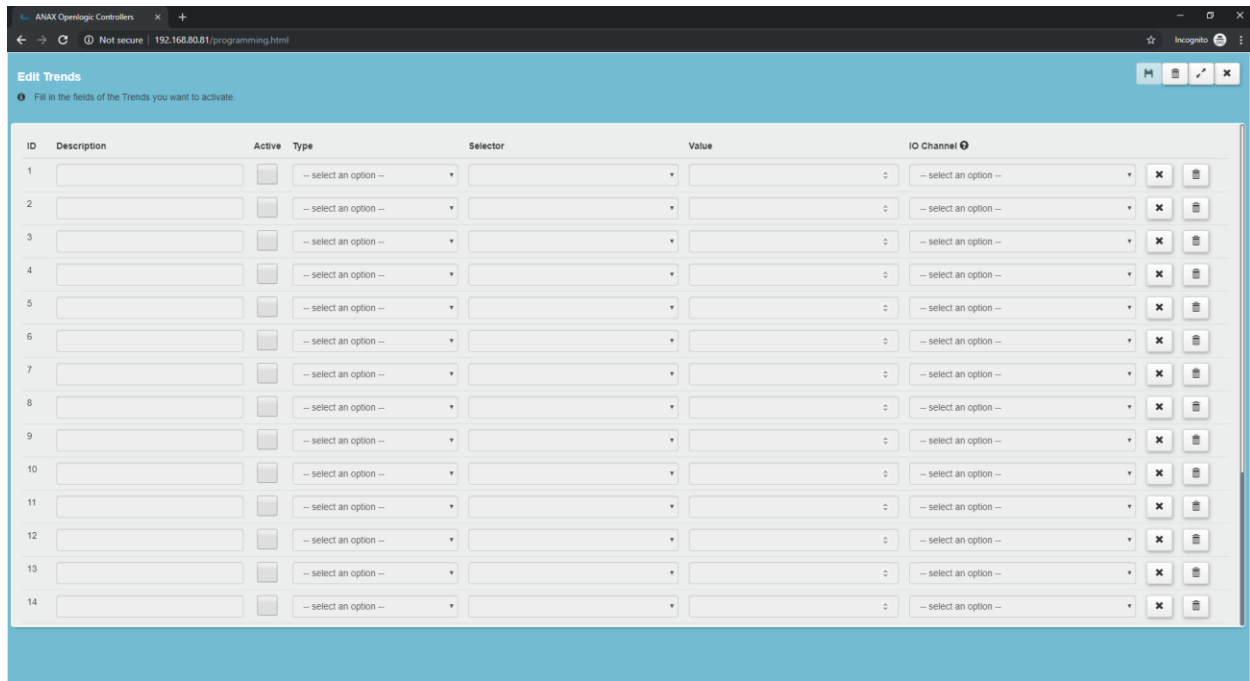
Trends is a data logging mechanism included in the uC's firmware. It creates data records (date, time and value) over time from user specified IO channels either at a fixed time interval or when certain conditions are met (e.g every 1 second, when the value of a channel of an IO equals a desired value e.t.c).

The 'Trends Configuration' button on the top right corner of each project's section (See 1 of image below) opens the modal with the settings related to the project's Trends handling.



Trends Configuration button

Each project has twenty (20) available Trend entries that are initially deactivated with no settings applied (See image below).

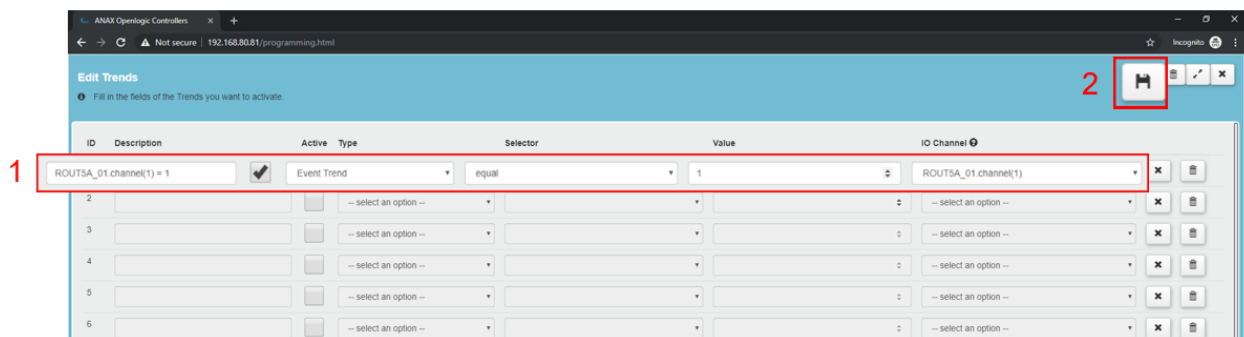


ID	Description	Active	Type	Selector	Value	IO Channel
1		<input checked="" type="checkbox"/>	-- select an option --			-- select an option --
2		<input type="checkbox"/>	-- select an option --			-- select an option --
3		<input type="checkbox"/>	-- select an option --			-- select an option --
4		<input type="checkbox"/>	-- select an option --			-- select an option --
5		<input type="checkbox"/>	-- select an option --			-- select an option --
6		<input type="checkbox"/>	-- select an option --			-- select an option --
7		<input type="checkbox"/>	-- select an option --			-- select an option --
8		<input type="checkbox"/>	-- select an option --			-- select an option --
9		<input type="checkbox"/>	-- select an option --			-- select an option --
10		<input type="checkbox"/>	-- select an option --			-- select an option --
11		<input type="checkbox"/>	-- select an option --			-- select an option --
12		<input type="checkbox"/>	-- select an option --			-- select an option --
13		<input type="checkbox"/>	-- select an option --			-- select an option --
14		<input type="checkbox"/>	-- select an option --			-- select an option --

Trends Configuration modal

The programmer can activate a Trend by clicking the related checkbox. He can set the settings for the specific Trend (See 1 of image below) and save the changes by clicking the 'Save' button on the top right corner. The 'Save' button is enabled upon making any change (See 2 of image below).

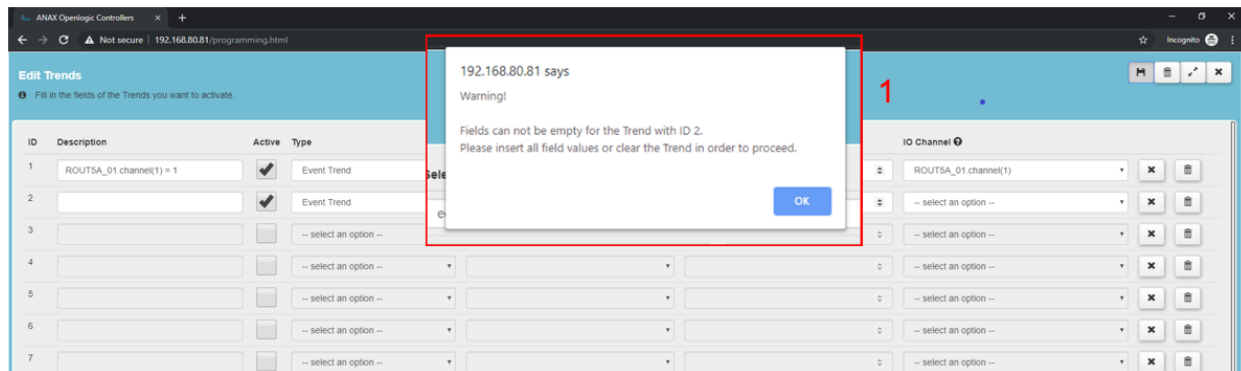
This 'Save' button is independent from the 'Save to board' button that refers to the project. Trends changes are saved instantly to the board for the specific project. This functionality gives the programmer the opportunity to change the project's Trends while a project is active.



ID	Description	Active	Type	Selector	Value	IO Channel
1	ROUTSA_01 channel(1) = 1	<input checked="" type="checkbox"/>	Event Trend	equal	1	ROUTSA_01 channel(1)
2		<input type="checkbox"/>	-- select an option --			-- select an option --
3		<input type="checkbox"/>	-- select an option --			-- select an option --
4		<input type="checkbox"/>	-- select an option --			-- select an option --
5		<input type="checkbox"/>	-- select an option --			-- select an option --
6		<input type="checkbox"/>	-- select an option --			-- select an option --

Trends Configuration modal

In order for a Trend to be accepted as active, all fields must be filled. If not, the programmer is informed upon clicking the 'Save' button and no changes are saved (See 1 of image below).

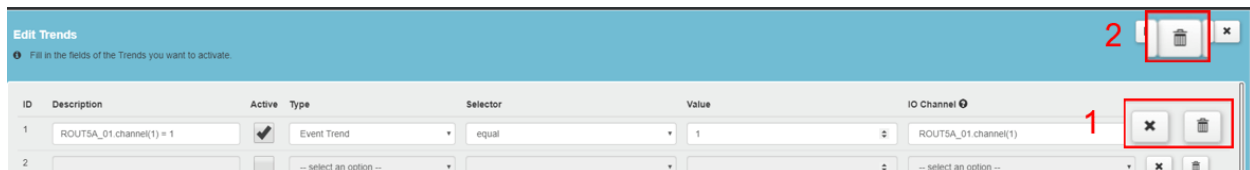


Trends Configuration modal

By unchecking a Trends' checkbox, the logging procedure is paused for the specific Trend and can be resumed by checking it again.

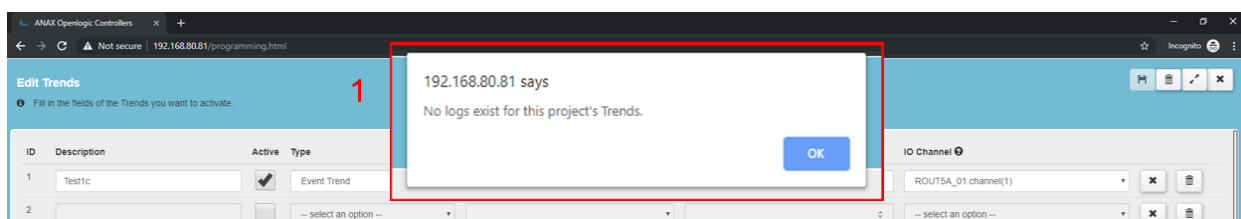
Two buttons exist on the right side of each Trend. The first button clears the selected Trend's settings and the second one clears instantly all existing logs for the specific Trend. (See 1 of image below).

The programmer has the ability to clear the existing logs of all Trends by clicking the trash bin button on the top right corner of the window (See 2 of image below).



Trends Configuration modal

If the logs of the specific project's Trends have been previously cleared by the uC, no action is performed and the programmer is informed accordingly (See 1 of image below).



Trends Configuration modal

When the programmer finishes with the declaration of the trends he wants to use, he can see the result in the run page which is explained in the manual [Help_For_Run_Page](#).

5 Reordering & misconfiguration cases

This section describes the system behavior of the Hydronic controller according to the physical IO connection to the RS485 communication line and analyses all possible cases of reordering or misconfiguration of the IO devices.

5.1 Preconditions

For all the following cases a correct system configuration is assumed. Mainly that the IO addressing has been executed successfully and thus the serial numbers appear in the IO Configuration panel.

In that case the IO Configuration declarations match exactly one-by-one the physically connected IOs. Considering that this IO configuration is static, with no further changes, the following cases are identified.

5.2 Cases Analysis

5.2.1 Bad or no communication

With an active project, the communication may be corrupted and lost at some IO device and beyond.

All the IOs, communicating or not, shall be set to their default deactivation (safety) values. The python code shall continue its execution, but without affecting any IO behavior during corrupted communication state.

The uC shall try to reconnect with the IOs and restore the communication, making multiple iterations through the communication bus, identifying each connected IO. The system shall provide a time frame of approximately 1-2 minutes for the missing IOs to reconnect and if they do so, then all the IOs' channels shall be reconfigured (activation values, channels, sensors, PWM, etc.).

In case the communication is not restored inside this time frame, the project shall get deactivated and the uC shall auto-reset and attempt only once to reactivate the project. Upon reactivation, if the communication is restored and no further problems exist (e.g. IOs misconfiguration), the project shall run normally.

In case that errors are detected, the project shall get deactivated again and the programmer must activate it manually through the UI.

5.2.2 Swapping IOs of same type

Independently of an active project or not, two IOs of the same type get swapped.

In that case, the system shall update automatically the IO configuration list and shall continue normally its operation.

5.2.3 Swapping IOs of different type

Independently of an active project or not, two IOs of different type get swapped.

As in the previous case, the system shall update automatically the IO configuration list and shall continue normally its operation.

5.2.4 Replacement of IO of same type

Independently of an active project or not, an IO gets replaced with a new IO of the same type.

In that case, the system shall update automatically the serial number of the old IO entry with the new one and shall continue normally its operation.

5.2.5 Replacement of IO of different type

Independently of an active project or not, an IO gets replaced with a new IO of different type.

In that case, the system shall detect an IOs **misconfiguration** status and shall alert the user.

5.2.6 Adding IOs at the end

Independently of an active project or not, additional IO devices are added at the end of the pre-existing connected IOs.

In that case, the system shall not be affected at all and shall continue its operation normally.

5.2.7 Adding IOs in between

Independently of an active project or not, additional IO devices are added among the pre-existing connected IOs.

In that case, the system shall consider this a reordering case and shall continue its operation normally.

Note: If the precondition at the beginning of this section about IO addressing is not met, the system understands that the user had not executed a successful IO addressing for the pre-existing connected IOs and this is not considered as a reordering case. Instead the system shall detect IOs **misconfiguration** status and alert the user accordingly.

5.2.8 Missing IOs

Independently of an active project or not, some IO devices are missing (total number of connected IOs is less than what is configured in IO Configuration).

i) When for any reason some devices in the bus are missing upon an IO addressing check, either manual, or automatically performed, the user shall be alerted for the specific **missing** IO devices.

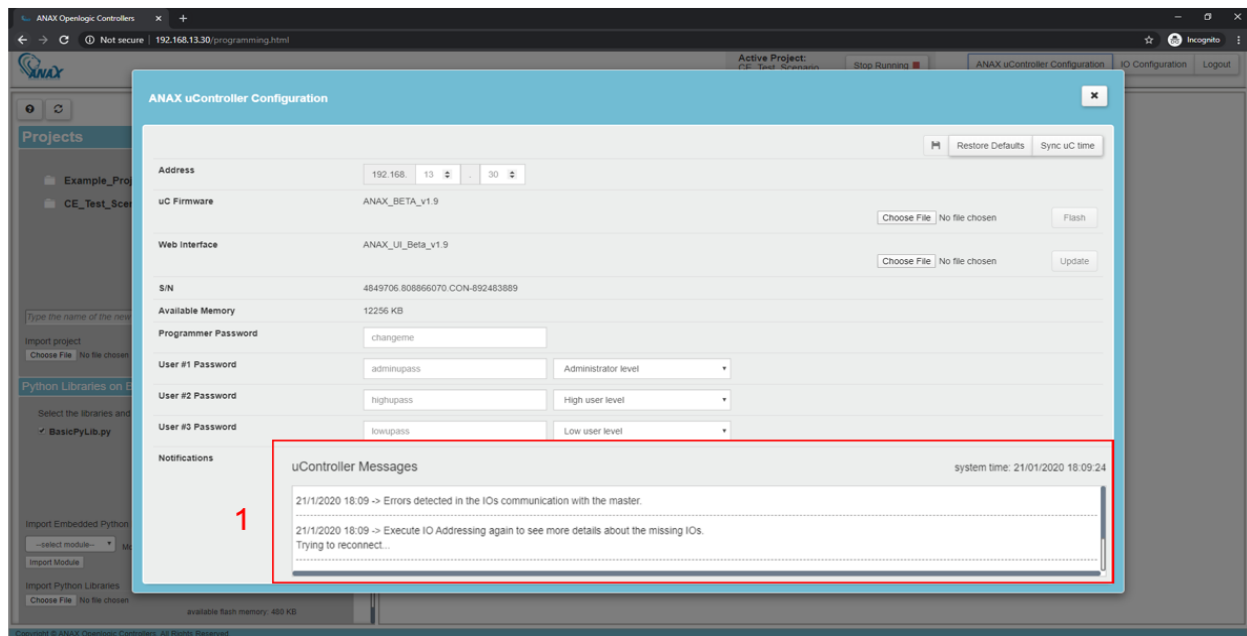
ii) When for any reason some devices in the bus are missing and in the same time others are not connected as expected in IO Configuration, the user shall be alerted only for the **misconfigured** IO devices, since the misconfiguration case is considered of higher importance and must be solved first. A second IO addressing will then inform for the missing ones.

5.3 Unsuccessful reordering

In case a reordering is not performed successfully the “Notifications” section of the uController Configuration window displays the appropriate messages (See 1 of Image below).

The same messages are displayed in the “uC Messages” section, in the index page of Hydronic controller.

Any previously active project is then deactivated and then only the System Programmer is able to configure and reactivate the project.



Notifications related to reordering

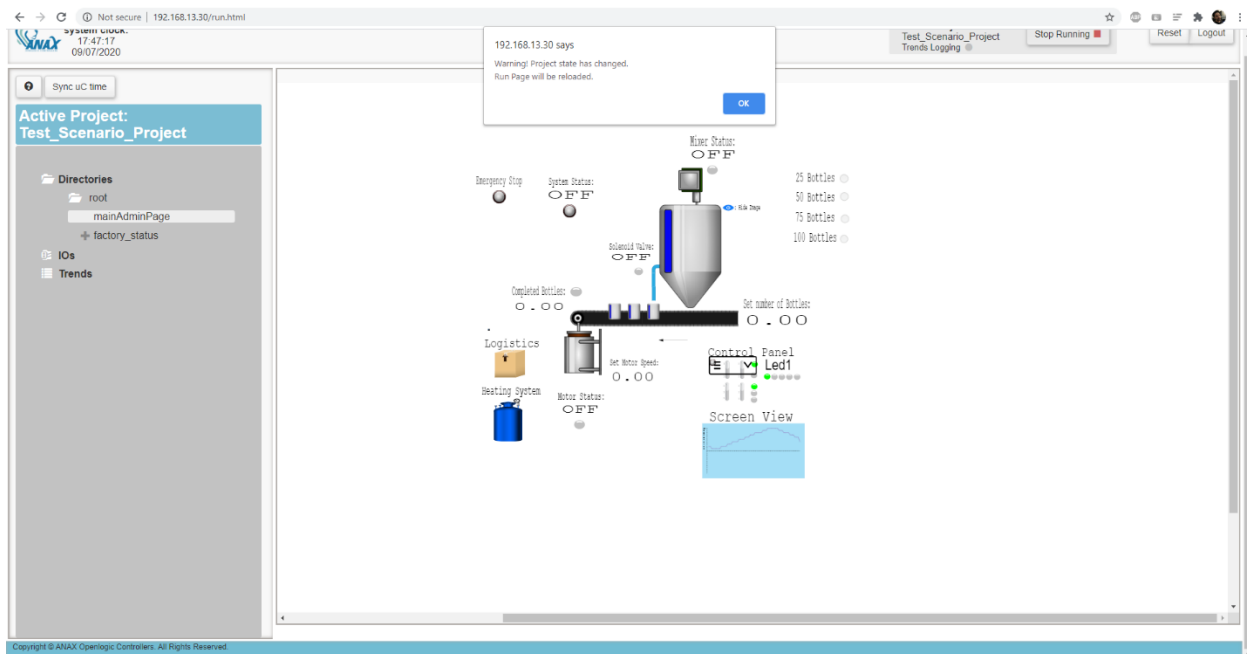
6 Programming/Run pages synchronization

In this section the functionality of pages synchronization is described depending on the browser and action taken.

Mozilla Firefox / Google Chrome

Case of programming and run page on the same window of the same browser.

1. Upon clicking 'Stop' button from run page the project gets deactivated. Run page is reloaded and the active programming page is automatically updated.
2. Upon clicking 'Start' button from run page the project gets re-activated. Run page is reloaded and the active programming page is automatically updated.
3. Upon clicking 'Stop' button from programming page the project gets deactivated. Programming page is automatically updated and run page gets reloaded (See 1 of image below).
4. Upon performing an activation from the programming page, the run page issues an alert message which informs the user that the project state changed and that the page will automatically reload in order to show the new state of the project.

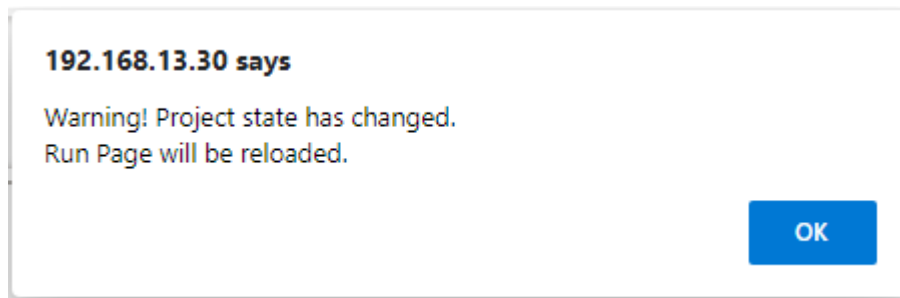


Run page upon detecting deactivation.

Microsoft Edge or Pages on different windows regardless of browser

In a cross-browser setup, a functionality has been added in the run page which constantly checks for a project status change. When the programmer changes the project status the run page automatically issues a warning and reloads (see 1 of Image below). If the project status change is performed from the run page, then the programming page issues a warning and reloads only on a crucial programmer action. Upon performing one of these crucial actions in the programming page, the user is informed when a change is detected on the state of the project. If the project status is not correct in the programming page, the crucial action is prevented and the page is reloaded in order for the settings to be updated. The actions considered as crucial for the system performance are:

1. Programming page
 - a. UI update
 - b. Firmware update
 - c. IO Addressing
 - d. Modification of IO Configuration
 - e. Python library deletion
 - f. Compile and Activation of a project
2. Run page
 - a. Stop/Start
 - b. Reset
 - c. Loading of Trends
 - d. Loading of Graphic pages
 - e. Loading of IOs



Run page automatic message issue for reload on a project status change in a cross browser setup.

7 Active Sessions

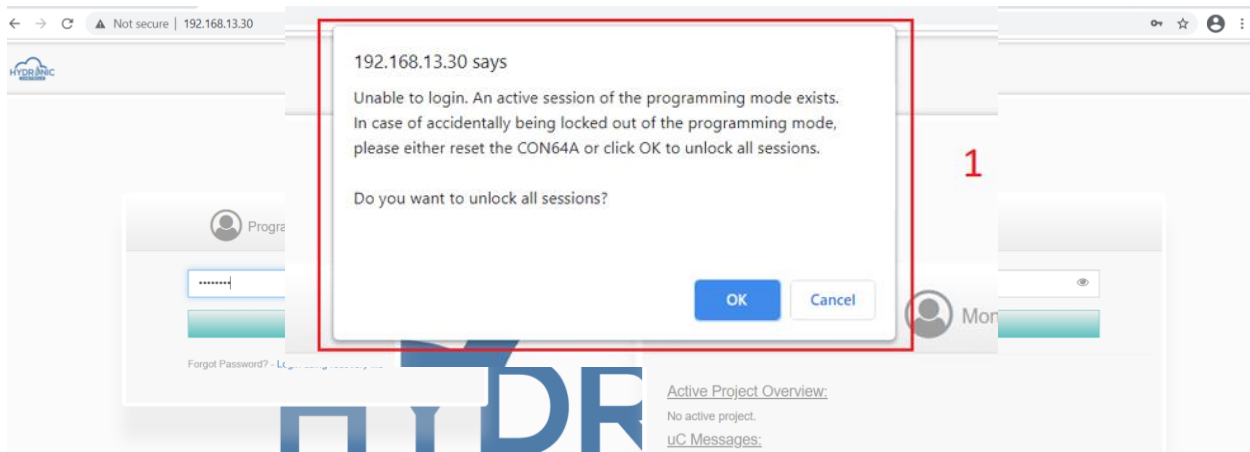
In this section the functionality that handles the existence of multiple UI open tabs is described. Restrictions have been applied to the number of allowed active sessions in order to protect and ensure the system's functionality. Regardless of the browser, window or computer only one active session is allowed per user level, hence, one session of programming page and one session of run page is allowed per user role.

A session is considered active when a user role performs a login and inactive upon logging out or closing the window or tab.

In case the MCU is powered off while a session is active, the session will be released upon power up of the MCU.

In case of trying to login either as a programmer or as a user while another active session exists, the appropriate message is displayed and login is prohibited (See 1 of image below).

The UI locks a session only if a user has logged in and the page has fully loaded.



Existing active session for programming page

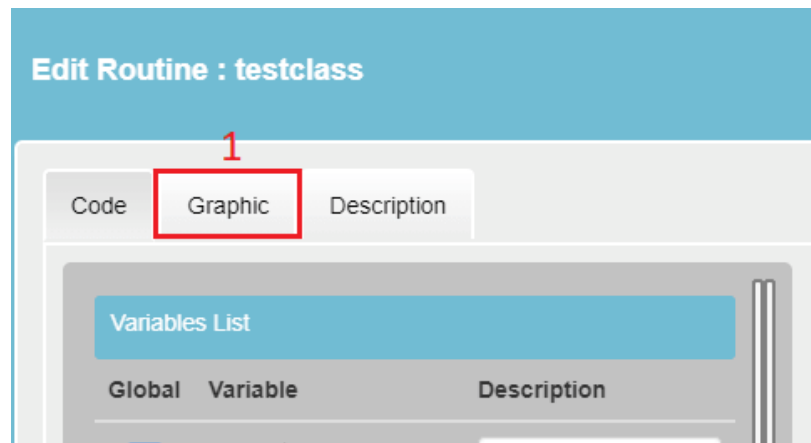
8 Graphics

In this section the graphic features of the programming page are described.

8.1 SCR and CON Routines Graphic Pages

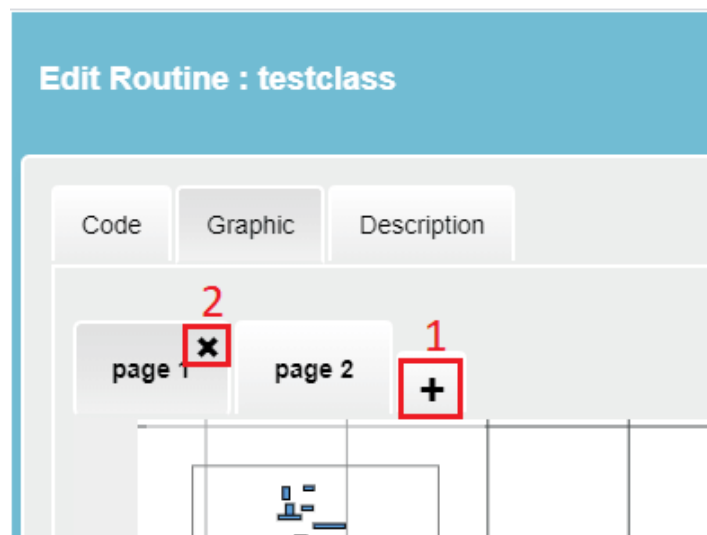
Each CON routine of a project includes its own graphic pages up to ten. An SCR routine is a routine to be used only for graphics design. The programmer can edit an SCR graphic page by clicking the third icon of the routine control section, as it is described in Routines Control Buttons section.

In case of a CON routine, the programmer can access the existing graphic pages by clicking the “Graphic” tab in the Routines Editor section (See 1 of image below).



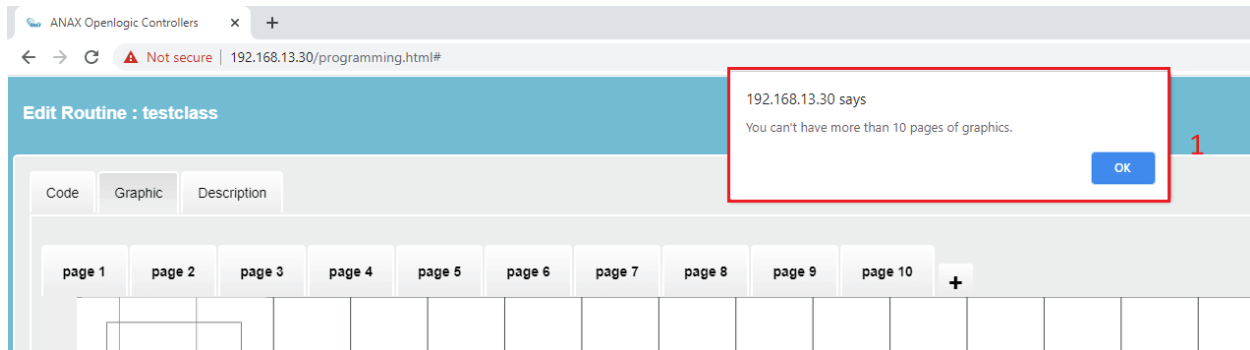
Edit Graphics in CON Routine

In a CON routine the user can add (See 1 of Image below) or delete (See 2 of Image below) graphic pages and edit each one of them independently.



Add and Delete Graphic Pages

The programmer can add up to 10 graphic pages in a CON routine, otherwise the programming page issues a warning (See 1 of Image below).

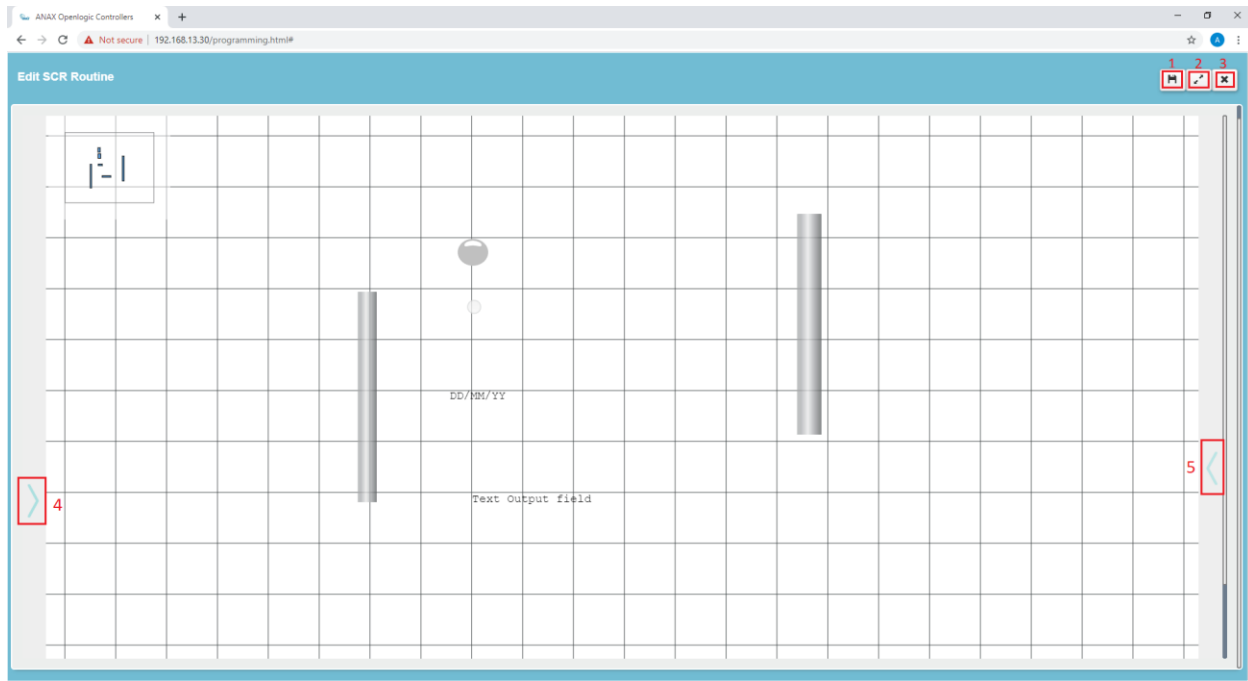


Up to 10 Pages are Allowed

8.2 Graphic Page Control Buttons

In a graphic page the programmer may see 3 buttons on the top right corner. By clicking the first button (See 1 of Image below), all inserted graphics are saved directly to the board. The saving procedure of graphics changes to the board is independent from the 'Save to Board' procedure which refers to the whole project. The second button (See 2 of Image below) minifies and restores the window and the third one (See 3 of Image below) closes the graphic page.

On the left and right side of the window there are two arrow buttons (See 4, 5 of Image below) which display the Left and Right-side menu of the graphic page.



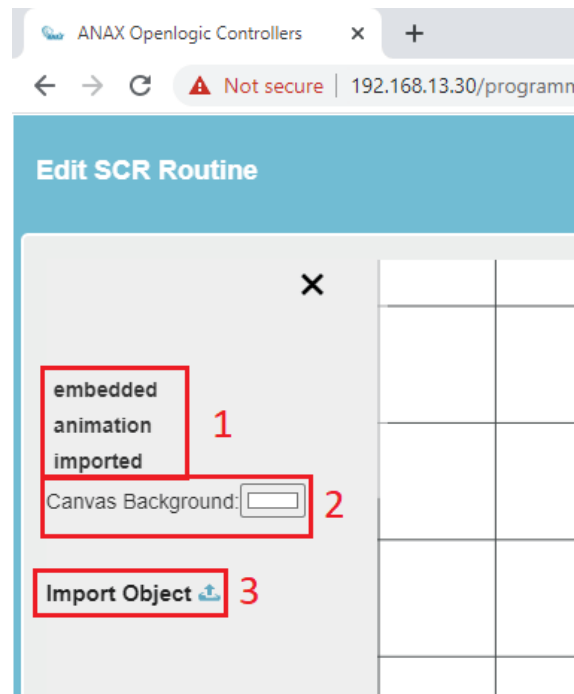
Graphic Page Control Buttons

8.3 Left Side Menu

The Left Side Menu contains the 3 categories of graphics, Embedded, Interactive and Imported objects (See 1 of Image below) which are described in 9.6 section. The programmer can see all the available graphics by clicking the desired category.

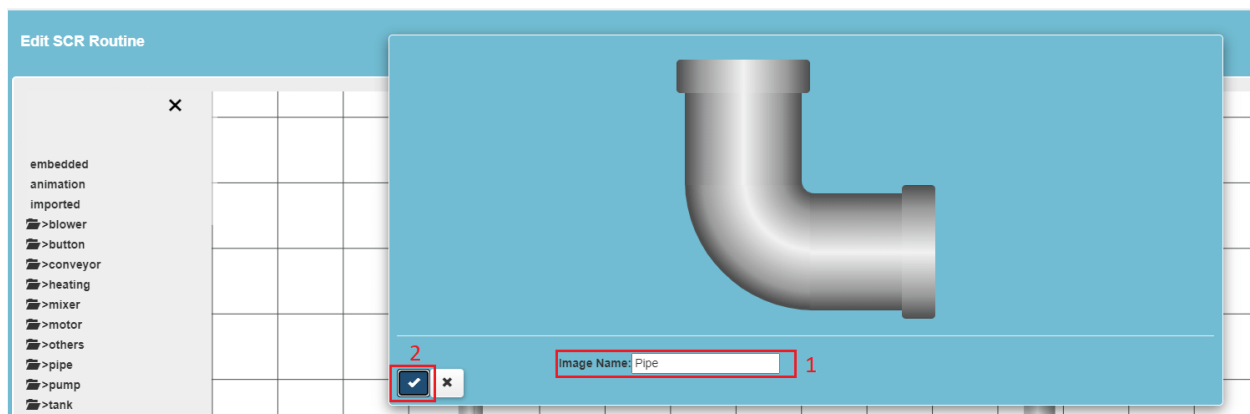
He is also able to change the background color of the canvas (See 2 of Image below).

Finally, the programmer has the ability to import images and use them as graphic objects on the canvas. By clicking the “Import Object” button (See 3 of Image below) the user is asked to browse and choose an image to Upload.



Left Side Menu Actions

A modal box will open and ask the user to define a unique name for the imported image (See 1 of Image below).



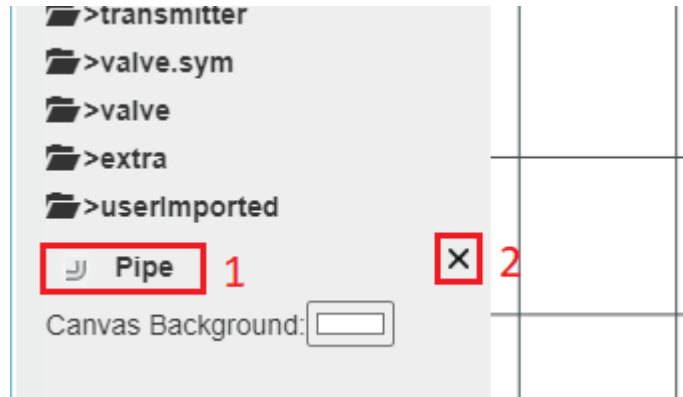
User Imports Object

After clicking the check button (See 2 of Image above), the image will appear in the left side menu in the section Imported/UserImported (See 1 of Image below).

He is now able to click on the imported image and it will appear on the canvas. All the main properties for the imported objects are available for this object too.

A delete icon is displayed next to the imported objects' name (See 2 of image below) so that the programmer may delete it if unnecessary and free up system's memory.

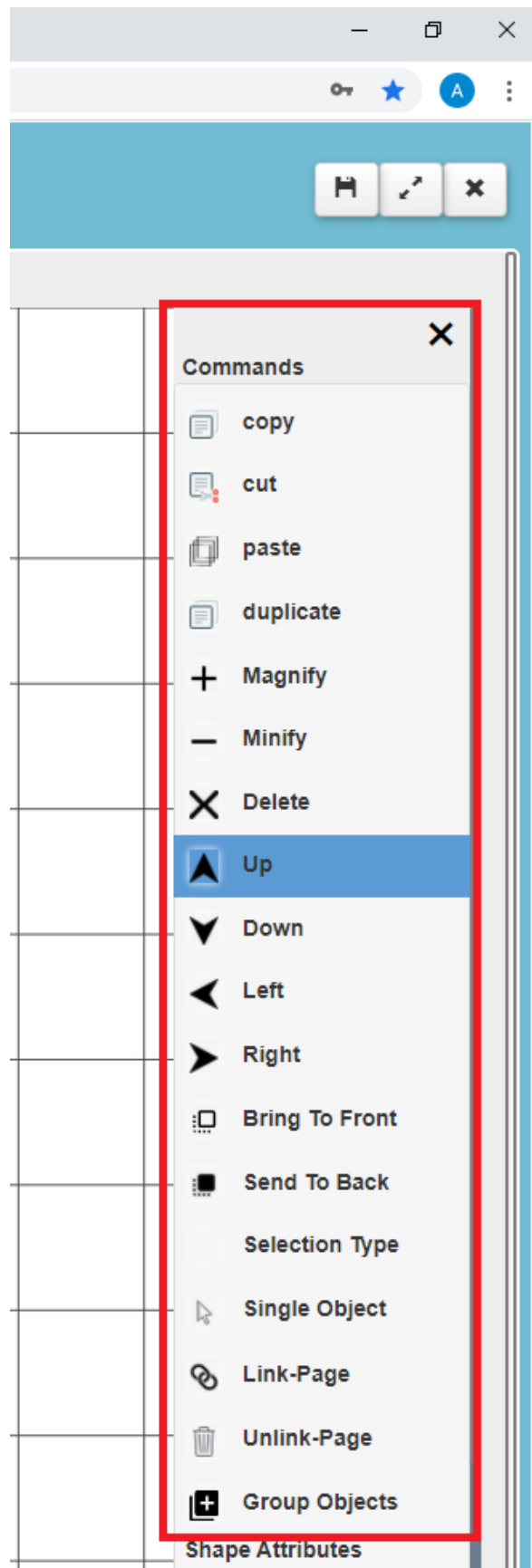
Upon import/delete of an image refresh the page if the changes do not take effect.



User-Imported Object

8.4 Right Side Menu

The Right-Side Menu contains all the available commands for the canvas' objects (See Image below). These commands have the same functionalities for all the embedded, interactive and imported objects which have been selected by the programmer.



*Right Side Menu***8.4.1 Copy, Paste, Cut, Delete, Duplicate**

These commands execute the basic functionalities of a graphic object. The *Paste* command works in the same or different canvas if the programmer has executed a *Copy* or a *Cut* command.

8.4.2 Magnify/ Minify

With these commands the programmer is able to change the size of a selected object

8.4.3 Up, Down, Left, Right

The programmer has the ability to move the selected object in the canvas.

8.4.4 Bring To Front/ Send To Back

In case of two or more objects overlap each other, the programmer is able to choose which one of them wants to bring in the front layer, using *Bring To Front* command, or send it in the back layer, using *Send To Back* command.

8.4.5 Selection Type

The *Selection Type* command is implemented in a way that it supports three different selection methods on the canvas' objects.

By default, the *Single Object* selection method is selected meaning that the programmer can select only one object at a time on the canvas with a left click on it.

Clicking either the *Selection Type* command or the displayed selection method e.g. *Single Object* changes the selection method to the next available.

The *Sticky Selection* method allows the programmer to select multiple objects on the canvas by left clicking on them.

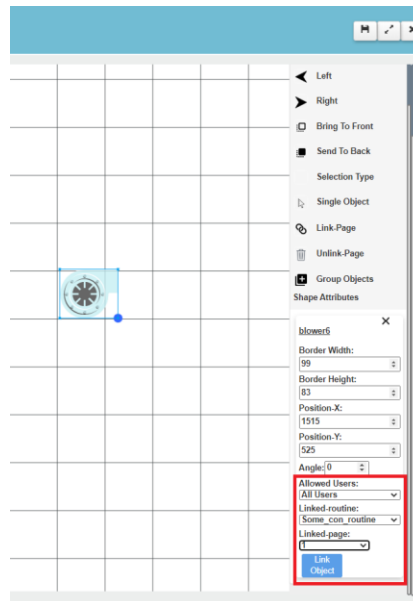
Finally, with the *Mouse drag* method the mouse can be dragged on the canvas in order multiple items to be selected inside a drawn rectangle.

8.4.6 Link Page

The Link/Unlink functionality allows in specific type of objects in SCR routines to connect with graphic pages of CON routines. So, Linking is only possible from SCR pages to CON pages. The user must follow the following procedure to successfully create a link.

Step1: From an SCR routine page and with the desired object to add a link selected, the user must press the Link-Page option from the right-side menu.

Step 2: After click a new section in the object card will be added (See Image below). The user must choose the routine to create a link to and then the page of this routine and press the “Link Object” button. If no error message appears, the Link was successful. (For the error messages see the Error notifications sections below).



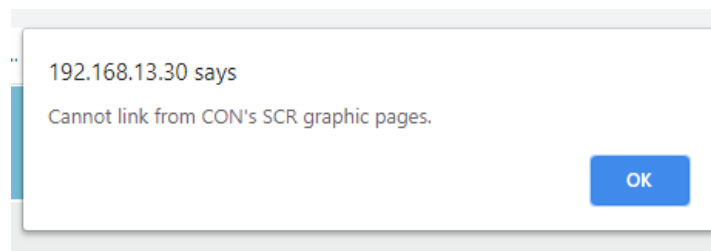
Step 3: The user can inspect the Link by double clicking on the object, both in programming & run page (See Image below).



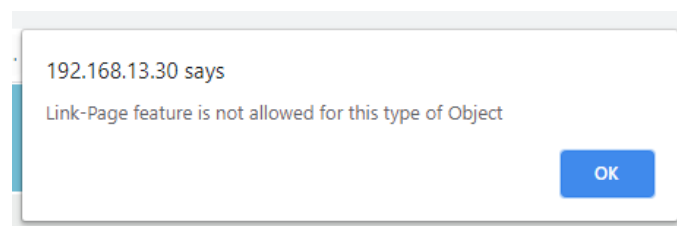
In order to Unlink the object, the user must select the object and press the “Unlink Page” option (See Image below).

Error notifications during connection:

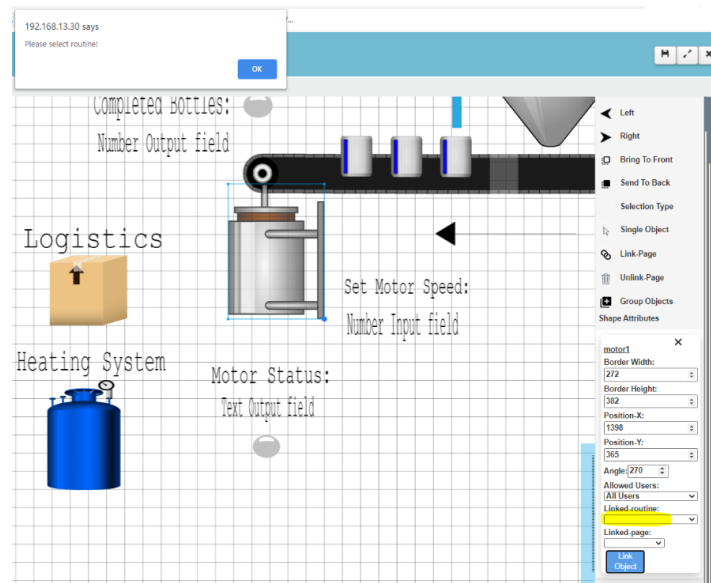
- 1) The Link-Page Feature is only allowed for graphic pages of SCR routines. In case the user is in a con routine and presses the “Link-Page” button from the right side menu, the following notification will appear:



- 2) The Link-Page Feature is only allowed for specific types of graphics. In case the user tries link a page with a non-allowed type of graphic, the following notification will appear:



- 3) The Link-Page Feature will be completed if the user has selected a valid graphic-routine and a valid graphic-page. In case the user presses the “Link Object” button, without selecting first graphic-routine and graphic-page, relative notifications will appear. Bellow appears the notification when no valid routine was selected:



8.4.7 Group Objects

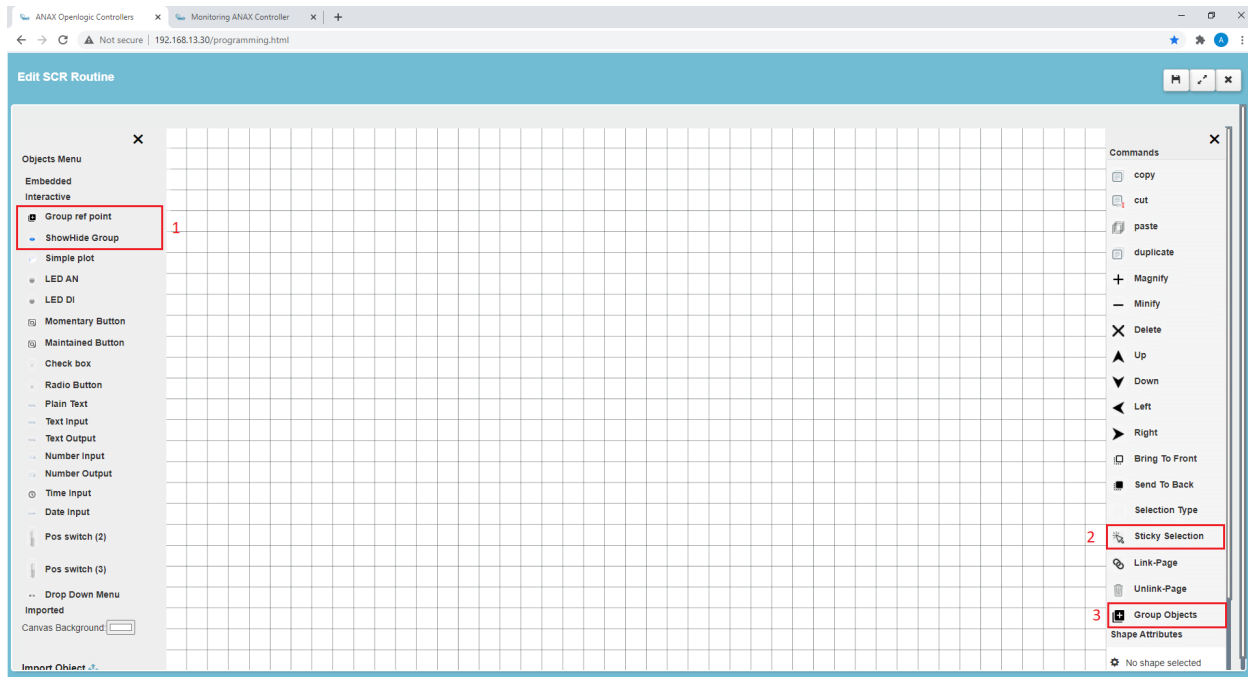
This command allows the programmer to group objects, so that he can move them in the canvas. The functionality of Group Objects is available and implemented ONLY using two object types, Group ref point and ShowHideGroup from the Interactive objects.

The user must follow the following procedure to successfully create a group.

Step 1: The user must insert in the canvas one of the objects that are mentioned above from the interactive objects list on the Left Side Menu (See 1 of Image Below).

Step 2: The user must select with an appropriate Selection Type method (Sticky Selection or Mouse Drag) the inserted interactive object (Step 1), as well as the objects he wants to group. (See 2 of Image below).

Step 3: He must click the *Group Object* command to complete the procedure (See 3 of Image below).



Group Objects

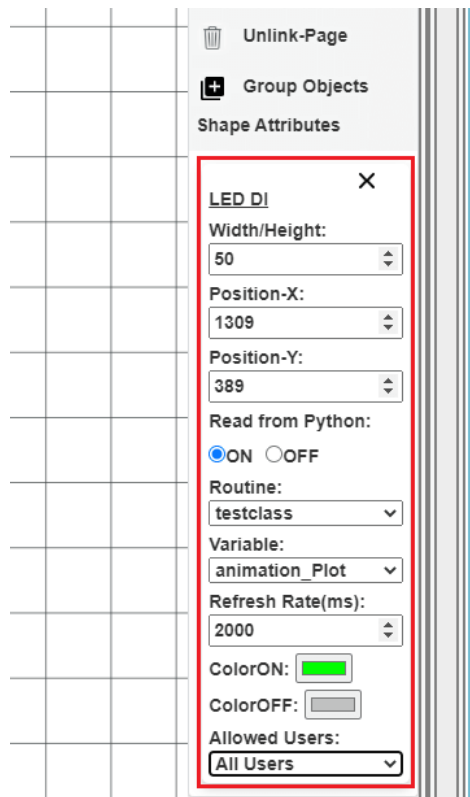
The user is now able to move the objects of the group by moving the interactive object (Step 1). If he tries to move any other object of the group, the rest of them are not affected.

Group ref point has the ability to move all objects of the group. ShowHide Group object has the extra functionality to set the group visible and invisible with a left click on the object.

To ungroup objects, the programmer must delete the object that handles the group (Group ref point, ShowHideGroup).

8.5 Object Info Card

When the programmer selects an object from the canvas, a card with information about the selected object is revealed in the Right-Side Menu (See Image below). The fields of the card vary depending on the type of the selected object. The programmer can make design and functionality changes in the objects from this object info card related to size, color, rotation, user privileges etc.

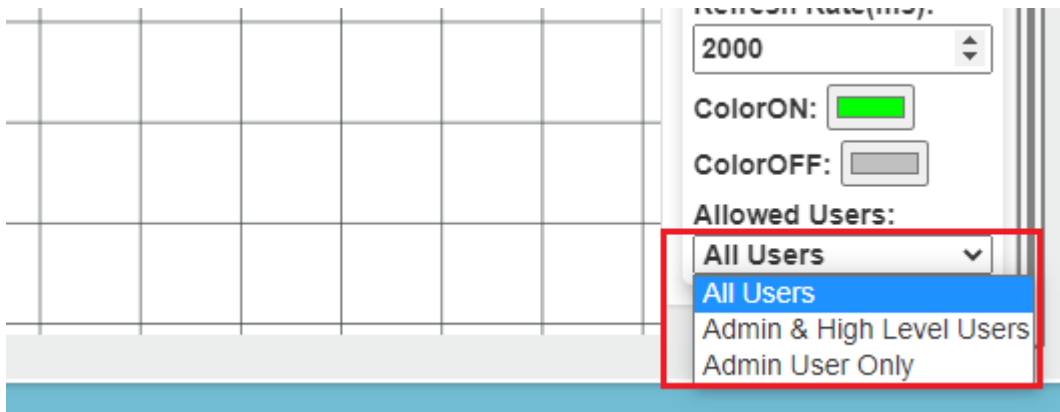


Object Info Card

For all objects the programmer may define specific Viewer user privileges for each object in the canvas among 3 available levels: 'All Users', 'Admin & High Level Users' and 'Admin User Only' (See Image below). The default option is 'All Users' meaning that all users may see the object in the canvas.

Additionally, ONLY for the Interactive Input type objects he may define Editor user privileges. The default option is 'Admin User Only' meaning that only the Administrator User Level is able to interact with these objects.

The Viewer and Editor privileges are defined per object separately even for objects that belong to the same group. This means that even when making a grouping object invisible to some user levels the rest objects are not affected.



User Privileges

8.6 Objects Categories

8.6.1 Embedded Objects

This category contains a variety of objects such as lines, crosses, corners, arrows etc. The programmer can adjust the size, angle, allowed users etc. using Right-Side Menu command and info card fields. The embedded objects are:

- solid tcross
- solid left-tcross
- solid-tline
- solid-tcorner
- left-tcross
- tcorner
- tline
- tlineCornerRight
- tlineCornerLeft
- darrow vertical
- darrow horizontal
- arrow vertical
- arrow horizontal
- full cross
- 3-side cross
- line
- corner

- dcorner
- dline
- 3-side dcross
- dcross
- 3-side bcross
- Bcross
- Bcorner
- Bline
- Left sq arrow button
- Right sq arrow button
- Left rd arrow button
- Right rd arrow button
- Frame 1
- Frame 2

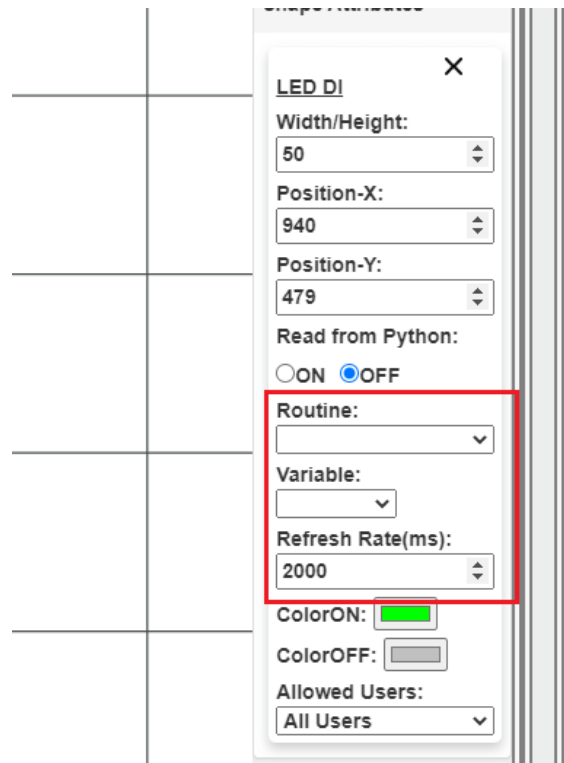
8.6.1.1 Arrow Buttons

The Arrow Buttons refer to the Left/Right sq arrow button and the Left/Right rd arrow button. These objects offer mainly navigation capabilities for the programmer in order to create connections between any graphic pages. The functionality is similar to the one when clicking any graphic page from the Project Explorer menu on the left.

The programmer has the ability to link an arrow button object with any routine's graphic page or any scr routine by selecting the desired routine and graphic page from Object Info card. In this case, the user in run page, can access the linked graphic page by clicking on the arrow button.

8.6.2 Interactive Objects

The interactive objects give the programmer the ability to connect them with python variables. This means that the user is able to display or change the value of a desired variable from his code. The programmer must select the routine and variable, as well as the refresh rate of the variable, from object info card (See Image below).



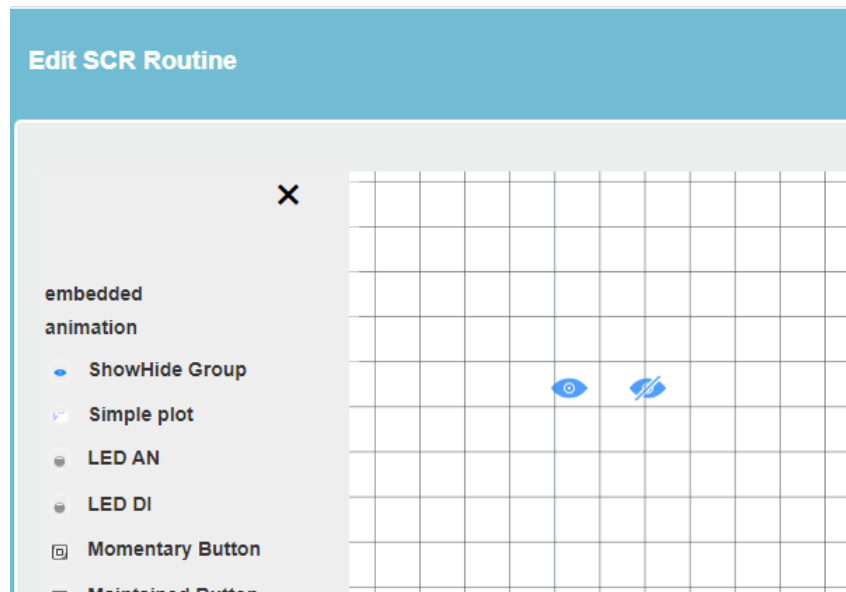
Interactive Objects Connected to Variables

8.6.2.1 Group ref point

Group ref point object is used for group object functionality. The procedure is described in [“Group Objects”](#) section.

8.6.2.2 ShowHide Group

This object gives the ability to a specified group of objects to be hidden/unhidden with a click. This functionality is only available for objects that are part of the group this object belongs to. This object is displayed as a blue eye icon (See Image below).



ShowHide Group object

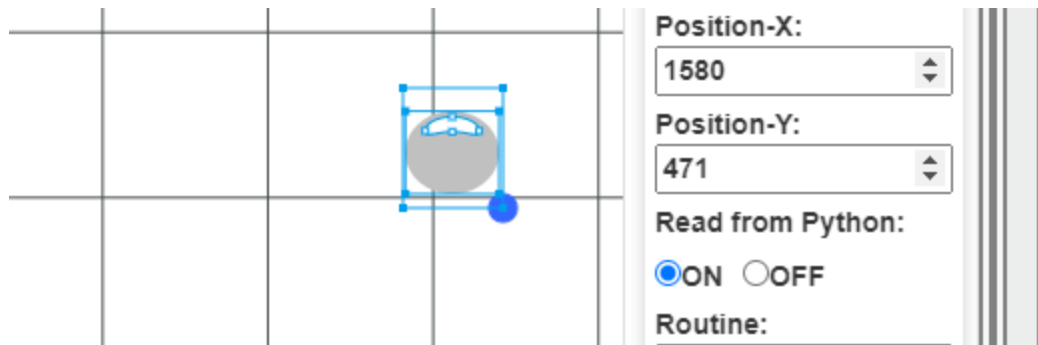
The programmer must select first all the desired items to be hidden along with this object and then from the **Commands menu** to click the option *Group Objects*. The procedure is the same with [Group Objects](#). A new group of all the selected objects is now created. By left click on the blue eye icon all the selected objects are hidden and the icon changes appropriately. Clicking again, the blue eye icon will make all the grouped items visible again.

8.6.2.3 LED DI

The 'LED DI' refers to digital values, hence, it shall be connected with a binary variable.

The color of the LED DI remains in ColorOFF selection during the design in the programming page.

In case the option 'Read from Python' is selected as ON (See Image below), the LED DI is set to color ON or OFF option in run page according to the value of the variable. The value of the variable is updated according to the selected refresh rate.



LED DI – Read from Python

The LED DI remains in ColorOFF selection in run page when the option 'Read from Python' is turned to OFF. In this case the LED has no functionality in run page.

8.6.2.4 LED AN

The 'LED AN' refers to analog values, hence, it shall be connected with a variable which takes values in a pre-defined range.

'Range (Min)' and 'Range (Max)' fields have been added to 'LED AN' object in order to calculate its min and max blinking frequency depending on the analog value readings.

The LED AN remains in ColorOFF selection during design in the programming page.

In case the option 'Read from Python' is turned to ON, the LED AN is blinking in run page according to the value of the analog variable, the calculated frequency and the colors selection. The value of the variable is updated according to the selected refresh rate.

8.6.2.5 Momentary Button

The Momentary Button shall be connected with a binary variable.

During design mode in the programming page the button has no functionality at all.

When in run page, the button press indicates the ON state and the release the OFF state. The initial state of the button upon loading, is always set to OFF, thus it is affecting the connected variable immediately. Upon clicking, a high value (1) will be set to the connected variable. Upon releasing, a low value (0) will be set to the connected variable. The colors of the button should match the ones selected by the programmer for each state.

Maintained Button

The *Maintained Button* shall be connected with a binary variable.

During design mode in the programming page the button has no functionality at all.

When in run page, the button can be clicked to toggle between ON/OFF state. The initial state of the button is decided based on the readings of the connected variable upon loading. Upon clicking for ON, a high value (1) will be set to the connected variable. Upon clicking for OFF, a low value (0) will be set to the connected variable. The colors of the button should match the ones selected by the programmer for each state.

8.6.2.6 Check Box

The Check box shall be connected to a variable of Boolean type that takes the values True/False.

During design mode in the programming page the box has no functionality at all.

When in run page, the box can be checked/unchecked for True/False state. The initial state of the box is decided based on the readings of the connected variable upon loading. When checked a high value (True) will be set to the connected variable. Upon unchecking, a low value (False) will be set to the connected variable.

8.6.2.7 Radio Button

The Radio button shall be connected to a variable of numeric type. The accepted values are either 0 for unchecked status or the number inserted in the field named “Value Checked” from the options menu in the programming page. In this way the run page will change the status of the Radio Button depending on the value of this field.

During design mode in the programming page the box has no functionality at all.

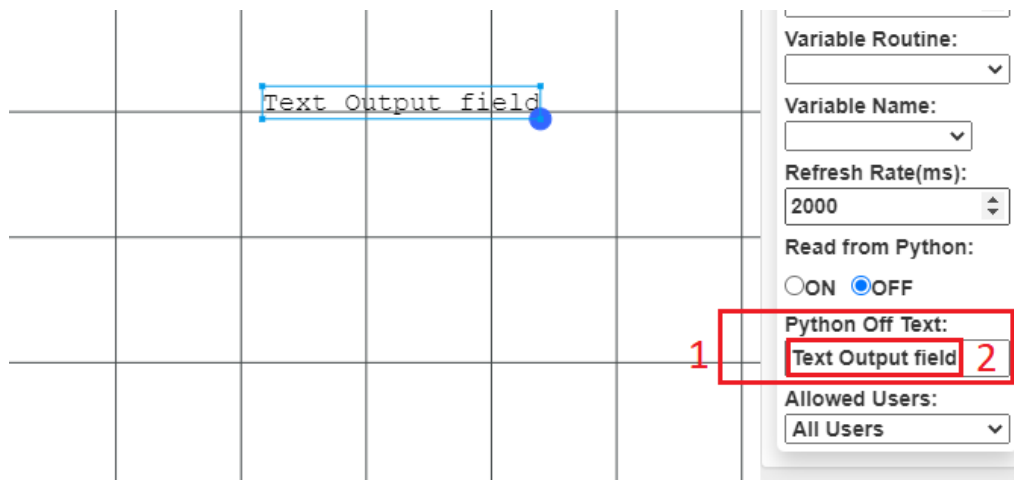
The initial state of the box in the Run page is decided based on the readings of the connected variable upon loading. When checked the value of the “Value Checked” field will be set to the connected variable. Upon unchecking, the zero value (0) will be set to the connected variable.

8.6.2.8 Text or Number Output

The Text or Number Output objects shall be connected with a variable of string or number type respectively.

In case the option 'Read from Python' is turned to OFF, no connection is made with the MCU and the 'Prgrmr Label' (See 1 of Image below) that has been typed by the programmer in the options menu is displayed in the object's position both for programming and run page.

The appropriate way to insert a new text is to select the whole text from the 'Prgrmr Label' and type the desired one (See 2 of Image below).



Text Output Field, Read from Python OFF

In case the option 'Read from Python' is turned to ON, the value of the connected variable is displayed in the run page. The value of the variable is updated according to the selected refresh rate.

8.6.2.9 Text or Number Input

The Text or Number Output object shall be connected with a variable of string or number type respectively.

In case the option 'Send to Python' is turned to OFF, no connection is made with the MCU and no input value can be set. The 'Prgrmr Label' that has been typed by the programmer in the options menu is displayed in the object's position both for programming and run page.

The appropriate way to insert a new text is to select the whole text from the 'Prgrmr Label' and type the desired one.

In case the option 'Send to Python' is turned to ON, the object is enabled in run page as a text or number input field. The initial value of the field, should match the existing value of the connected variable upon loading. By selecting the object, the user in the run page can type a new valid value and apply it to the variable connected to the specific object. The value of the variable is updated according to the selected refresh rate.

8.6.2.10 Time/Date Input

The Time or Date Input object shall be connected with a variable of string type that stores the time or date in the following formats respectively, hour: minutes: seconds or date/ month/ year. An example of such a variable usage may be seen below (See Image below). In this case the variable MCUTime gets content from the graphics and is compared with appropriate strings in the code. It could be also the opposite that these strings get content from the graphics and compared to the system time.

```

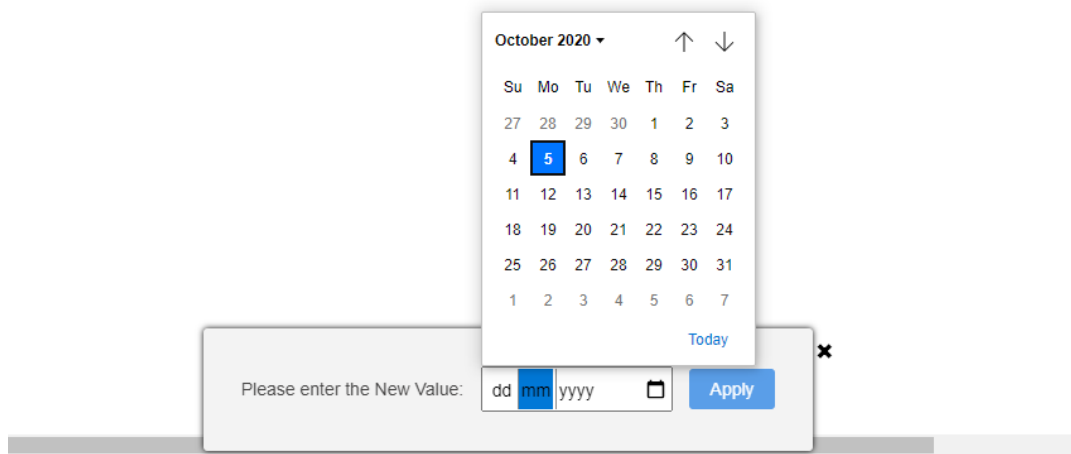
10 # Out Variables: List your Out Variables here
11 # <name :: description>
12 #
13 # =====
14 # Please follow the given template in order to ensure your code is working.
15 # =====
16
17 class Graphics_Class:
18     # This is your main class. You can add variables and methods.
19
20     # =====
21     # Variables Section
22     # =====
23     # Please initialize your variables after this line, e.g. x = 0
24
25     TimeOfActivation = '14:30:00'
26     TimeOfDeactivation = '22:30:00'
27     MCUTime = 0 # the MCUTime is connected to a 'Time' graphic object
28
29     dout8 = 1
30
31     def run(self):
32
33         # =====
34         # Code Section
35         # =====
36         # Your application must be written here. Add all your code.
37         # You can add your own methods to be used inside the run() routine. Add as many as you wish.
38         # Please insert your code after this line.
39
40         # the MCUTime is loaded in the format of hours:minutes:seconds
41         if self.MCUTime == self.TimeOfActivation:
42             # activates channel 8 of DOUT8A when the time of MCU will be equal to 14:30:00
43             self.dout8 = 1
44         elif self.MCUTime == self.TimeOfDeactivation:
45             # deactivates channel 8 of DOUT8A when the time of MCU will be equal to 22:30:00
46             self.dout8 = 0
47
48         # enddef
49
50     # endclass
51

```

Activation/Deactivation of a channel depending on the MCU time.

In case the option 'Read from System' is turned to OFF, the default Value 'DD/MM/YYYY and HH:MM:SS' is displayed in the object's position both for programming and run page. In this case the user is able to insert the desired time or date from the run page (See Images below).

21/10/2020



October 2020

Su	Mo	Tu	We	Th	Fr	Sa
27	28	29	30	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	31
1	2	3	4	5	6	7

Today

Please enter the New Value: dd/mm/yyyy

Apply

Date Input

08:00:00

Please enter the New Value:

08	00	00	AM
09	01	01	PM
10	02	02	
11	03	03	
12	04	04	
01	05	05	
02	06	06	

🕒
Apply

✕

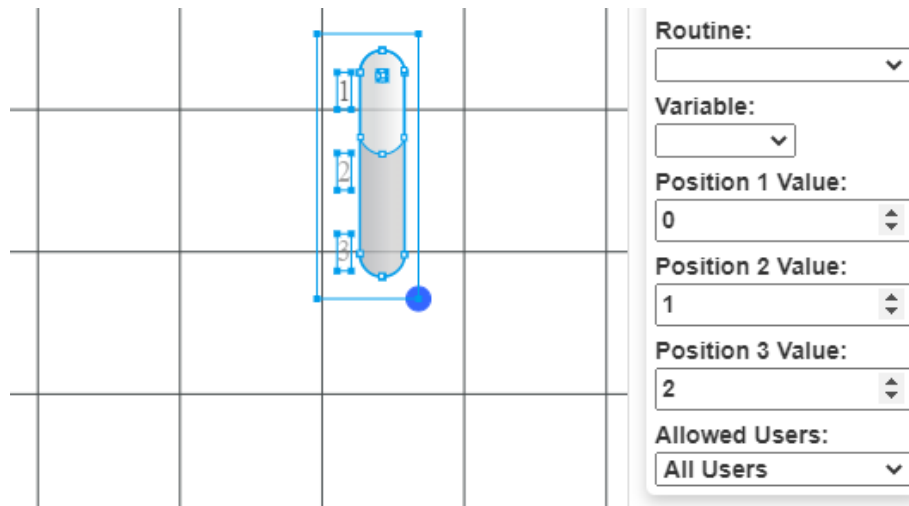
Time Input

In case the option 'Read from System' is turned to ON, the actual time or date of the MCU will be displayed in run page and, will be updated according to the selected refresh rate. These values will then be set to the connected variable.

8.6.2.11 Position Switch (2) & (3)

The Position Switch object shall be connected to a variable, which has at least 2 different values in case of Position Switch (2) or 3 different values in case of Position Switch (3). The User can set these values (levels) during design (See Image below). The switch changes state by clicking on the objects' image or when the value of the connected variable is automatically updated from the MCU. Depending on its position the switch assigns the selected value to the connected variable.

Both switches have also an UNDEFINED state which appears only when the connected variable has other value than the defined levels.



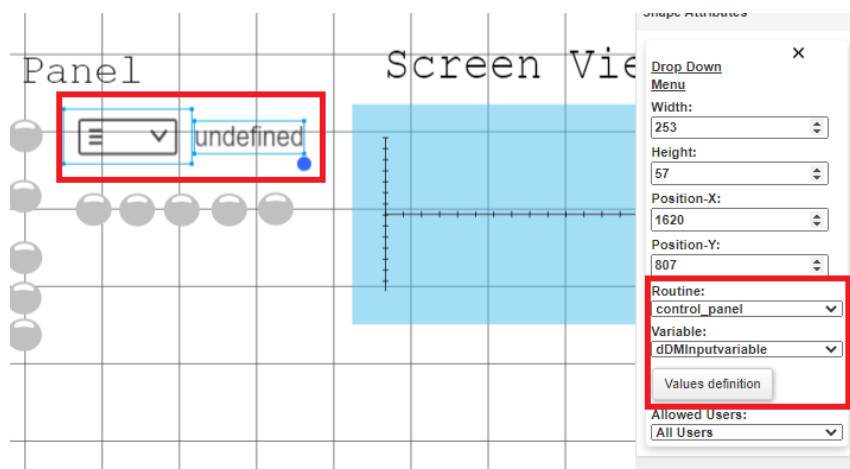
Setting Values for Position Switch (3)

8.6.2.12 DropDown Menu

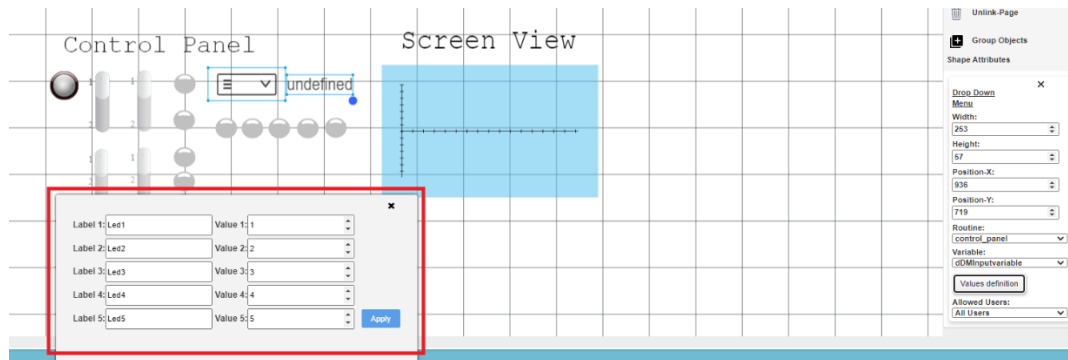
The DropDown Object shall be connected to a variable in which up to five 5 different values may be assigned. The programmer shall configure for each dropdown selection the displayed label and the value to be assigned to the connected variable.

In order for the user to assign labels to values must follow the following procedure:

Step1: From the programming page, the user must select the drop-down menu and after the selection of the Routine and the Variable, must press the “Values definition” button.



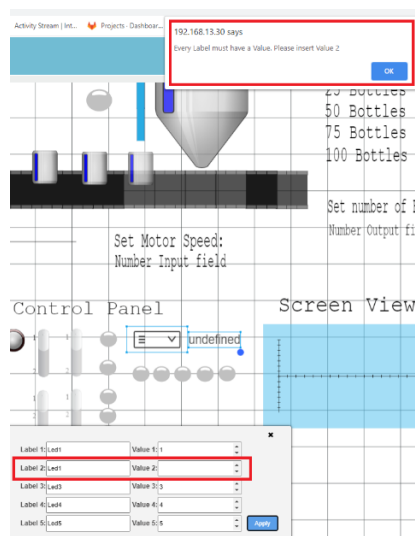
Step2: The user must insert the desired labels and values to the appropriate window and then press the “Apply” button.



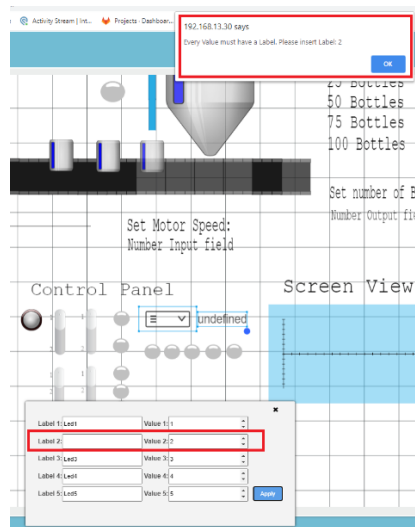
If the user clicks outside of the window or clicks on the close button the window will close **without saving the values or the labels**. After clicking on the Apply button, a number of checks are implemented. If everything is ok, the window will close without any message and the procedure is complete.

Below are the possible errors during the insertion of the values and labels.

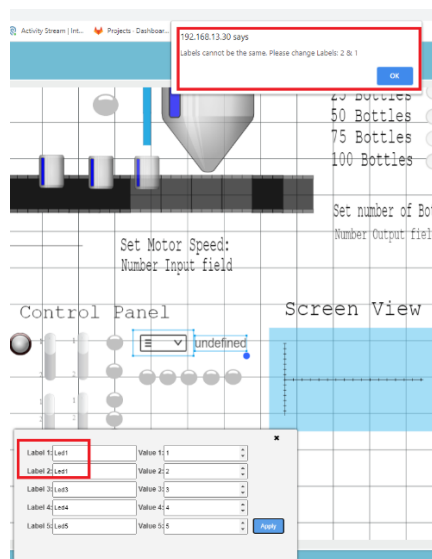
1) If there is a label without value, the following error message will appear:



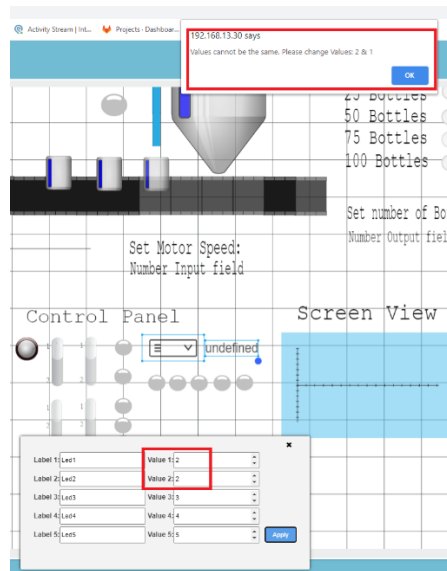
2) If there is a value without label, the following error message will appear:



3) If at least 2 labels are identical, the following error message will appear:

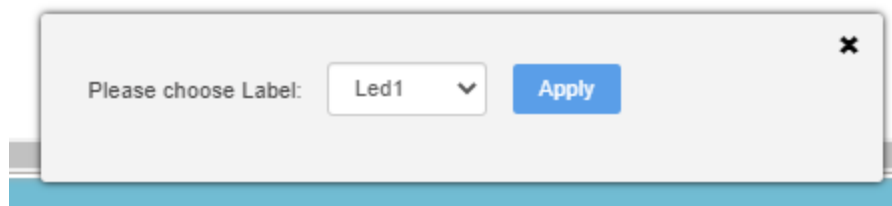


4) If at least 2 values are identical, the following error message will appear:



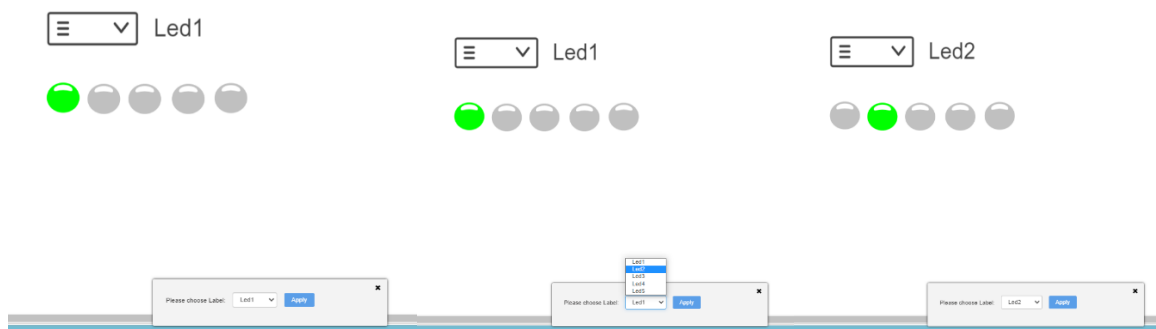
In the run page, the user can choose one of the labels by following the following procedure:

Step 1: The user must select the drop-down menu and the following window will appear.



Step2: The user can choose one of the labels and press the “Apply” button. If the user clicks outside of the window or clicks on the close button the new label **will not be set**.

The following 3 images shows the procedure:



The displayed label of the Drop-Down Menu has also the ability to change automatically in case the connected variable changes (i.e. from the python code). In case the connected variable has a value different than the 5 predefined values, the displayed label shows the word “Undefined” (state Undefined).

Upon initialization the displayed label of the Drop-Down Menu starts with state “Undefined” till it is initialized with the connected variable value.

8.6.2.13 Simple Plot

The Simple Plot Object shall be connected to a variable of arithmetic type in order to graphically display its value in time.

The user may change the background color of the plot from the field “Background Color” (See 1 of Image below). The default color is white.

The user may choose whether axis will be visible in the “Show Axis” field (See 2 of Image below).

The user has the option to choose the number of the samples by inserting a number in the History Tracks field (See 3 of Image below).

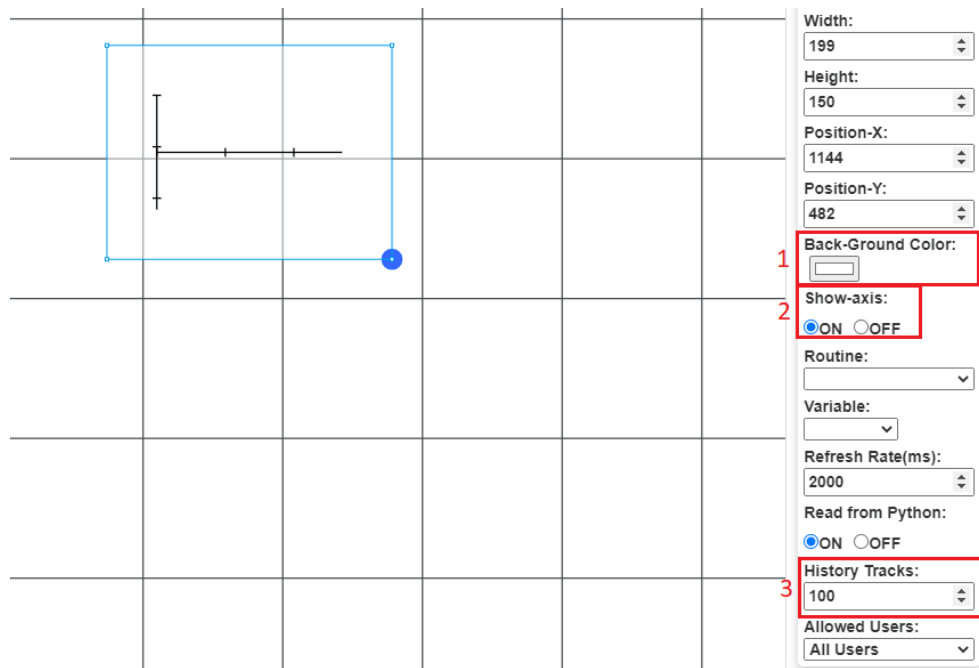
The user may also specify the refresh rate from the according field.

The user may choose whether the plot will be connected to a python-variable.

In case the option ‘Read from Python’ is turned to OFF, no connection is made with the MCU and the Plot object remains unchanged.

In case the option ‘Read from Python’ is turned to ON, the value of the connected variable is displayed in the run page. The values of the variable are depicted to the left side of the Plot.

The curve of the Plot diagram adjust dynamically in every new value that is being inserted and rescales the X-axis and the Y-axis automatically. This way the form of the curve depends of the number of the samples that is being stored.

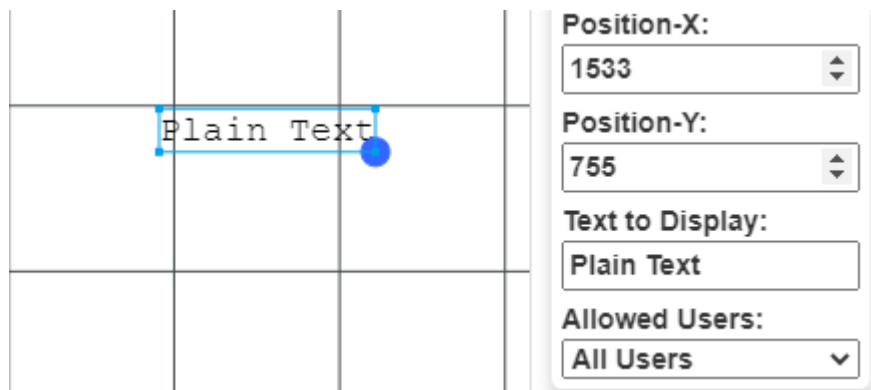


Customize Animation Plot

8.6.2.14 Plain Text

Plan Text is the only animated object that is not connected to any variables.

The programmer can use this object to display texts in his canvas by typing in 'Text to Display' field (See Image below). The appropriate way to insert a new text is to select the whole text from the 'Text to Display' field and type the desired one.

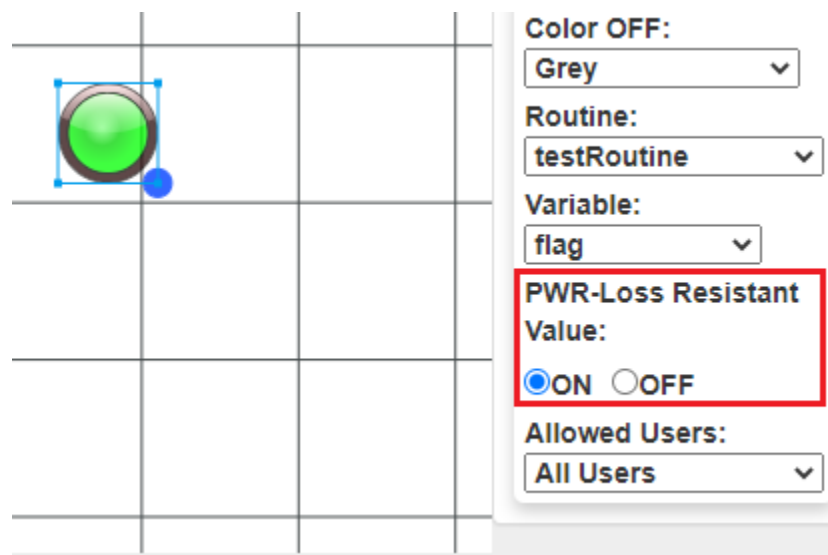


Plain Text

8.6.2.15 Power loss resistance values for Run page

For every interactive object of input type, there is a mechanism for saving any changes made in the graphic pages in run page by the Administrator User Level. This feature stores the value of the connected variable in order to restore it in a possible power-cycle of the uController.

By default, all input animation objects have the PWR-Loss Resistant Value set to ON. The programmer is responsible to indicate the objects that does NOT need to be power loss resistant during design. He has the ability to activate and deactivate this mechanism for every input type object as seen in the Image below.



Power Loss Resistance Button

When the programmer sets the PWR-Loss Resistant Value radio button to OFF, any previously stored value of the connected variable is deleted.

This mechanism is not object related rather it is variable related and indicated to the system the value of the variable to restore.

During restore of the variable's value the stored value is sent directly to the running code. In case multiple graphic objects are connected with the same variable, a deactivation of this mechanism for one object will affect all the rest objects.

For example, four radio buttons are used to indicate four different values of a variable. All the button by default were PWR-Loss Resistant and the project has been Compiled and Activated at least once. Let's say the radio button values are:

- None
- 20

- 50
- 100

all connected with the variable “RetriesSelection”.

In case that the programmer decides to make ONLY the first radio button non PWR-Loss Resistant (set to OFF) the system will remove the value of the “RetriesSelection” variable from the restore process. The system will restart and read the default value in the python code for the “RetriesSelection” variable.

In case though the Administrator User Level selects a value 20,50 or 100 and because the programmer has not made these objects non PWR-Loss resistant the selected value for the object will be stored again to the restore process.

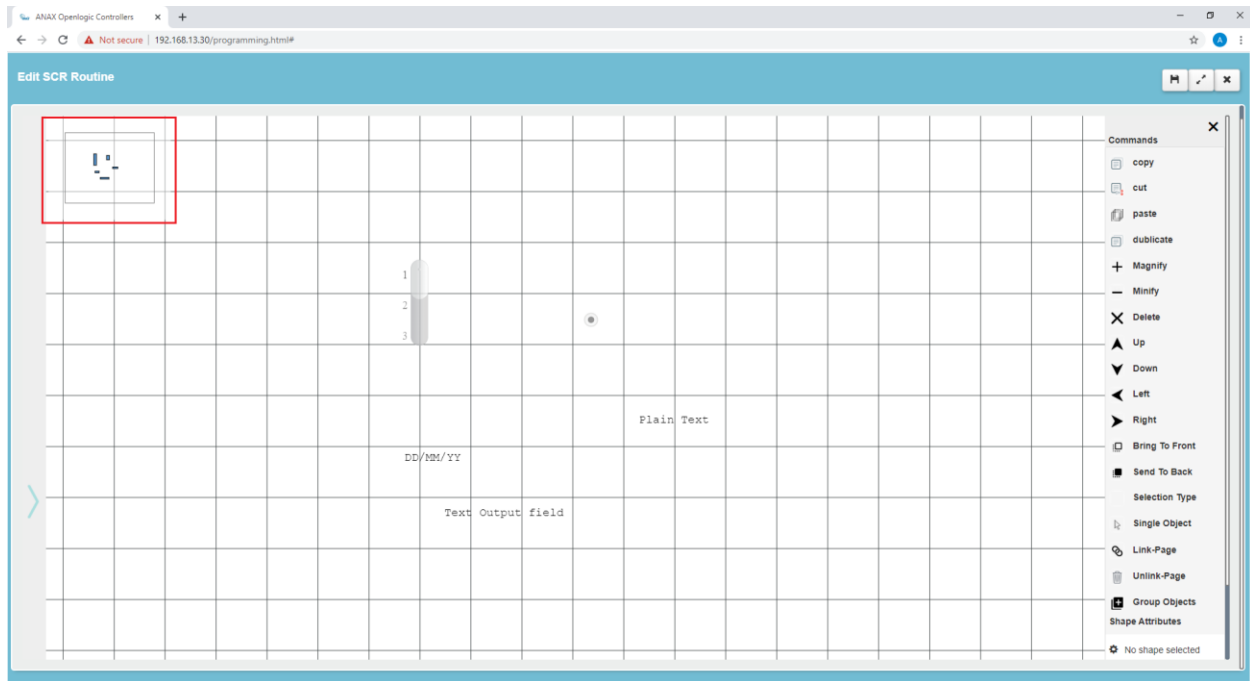
So, the change on the programmer’s side shall be made for all the objects connected with the specific variable.

8.6.3 Imported Objects

Imported Objects have the exact same functionalities with embedded objects. This category contains a variety of useful objects such as blowers, motors, tanks, valves etc.

8.7 Mini Map

Mini Map is a minimized view of canvas in order to give the programmer a quick overview of the canvas’ contents. The inserted objects are displayed as blue elements in mini map. The programmer is able to see if there are objects outside his current view in the canvas. The user cannot interact with mini map at all.



Mini Map

9 Python modules

In this section the python modules and the functions supported are listed. For more detailed documentation for the micropython, please refer to <http://docs.micropython.org/en/v1.10/>

Python Module	Functions and Constants
Builtinfunctions [1]	abs(), all(), any(), bin(), class bool, class bytearray, class bytes, callable(), chr(), classmethod(), compile(), class complex, setattr(obj,name), class dict, dir(), divmod(), enumerate(), eval(), exec(), filter(), class float, class frozenset, getattr(), globals(), hasattr(), hash(), hex(), id(), class int, classmethod from _bytes(bytes,byteorder), to _bytes(size,byteorder), isinstance(), subclass(), iter(), len(), class list, locals(), map(), max(), class memoryview, min(), next(), class object, oct(), ord(), pow(), property(), range(), repr(), reversed(), round(), class set, setattr(), class slice, sorted(), staticmethod(), class str, sum(), super(), class tuple, type(), zip()

array [2]	class array.array(typecode[,iterable]), append(val), extend(iterable)
cmath [3]	cos(z), exp(z), log(z), log10(z), phase(z),polar(z), rect(r,phi), sin(z), sqrt(z), e, pi
gc [4]	enable(), disable(), collect(), mem_alloc(), mem_free(), threshold()
math [5]	acos(x), acosh(x), asin(x), asinh(x), atan(x), atan2(y,x), atanh(x), ceil(x), copysign(x,y),cos(x), cosh(x), degees(x), erf(x), erfc(x), exp(x), expm1(x), fabs(x), floor(x), fmod(x,y), frexp(x), gamma(x), isfinite(x), isinf(x), isnan(x),ldexp(x,exp), lgamma(x),log(x), log10(x), log2(x), modf(x), pow(x,y), radians(x), sin(x), sinh(x), sqrt(x), tan(x), tanh(x), trunc(x), e, pi
sys [6]	exit(retval=0), argv, byteorder, implementation, maxsize, modules, platform, version, version_info
collections [7]	deque(iterable, maxlen[, flags]), deque.append(x), deque.popleft(),namedtuple(name, fields), OrderedDict(...)
micropython [8]	const(expr), opt_level([level]), alloc_emergency_exception_buf(size), mem_info([verbose]), qstr_info([verbose]), stack_use(),heap_lock(),heap_unlock(),kbd_intr(chr), schedule(func, arg)
struct [9]	calcszize(fmt),pack(fmt, v1, v2, ...), pack_into(fmt, buffer, offset, v1, v2, ...), unpack(fmt, data), unpack_from(fmt, data, offset=0)
errno [10]	EEXIST, EAGAIN etc, errorcode
binascii [11]	hexlify(data[, sep]), unhexlify(data), a2b_base64(data),b2a_base64(data)
heapq [12]	heappush(heap, item), heappop(heap),heapify(x)
random [13]	getrandbits(num_bits), seed(value)
re [14]	compile(regex_str[, flags]), match(regex_str, string), search(regex_str, string), sub(regex_str, replace, string, count=0, flags=0), DEBUG, regex.match(string), regex.search(string), regex.sub(replace, string, count=0, flags=0), regex.split(string, max_split=-1), match.group([index]), match.groups(), match.start([index]), match.end([index]), match.span([index])
time [15]	localtime([secs]), mktime(),ticks_add(ticks, delta), ticks_diff(ticks1, ticks2), time()
ctypes [16]	class struct(addr, descriptor, layout_type=NATIVE), LITTLE_ENDIAN, BIG_ENDIAN, NATIVE,sizeof(struct, layout_type=NATIVE), addressof(obj), bytes_at(addr, size), bytearray_at(addr, size), UINT8, INT8, UINT16, INT16, UINT32, INT32, UINT64,INT64, FLOAT32, FLOAT64, VOID,PTR, ARRAY
core	processor_clock()