ctrlX - CORE

- *Using Recipes in ctrlX*
- *- What is a recipe?*
- *- Recipe Control using WebVisu*
- *- Recipe Control using PLC Modules*
- *- USB control on a ctrlX Core (X3 & X7)*
- *- Nodered for USB control*
- *- Copy of the recipe PLC directory to the USB and vice versa*

Jordi Laboria (DCET/SLF4-ES) - V0.1

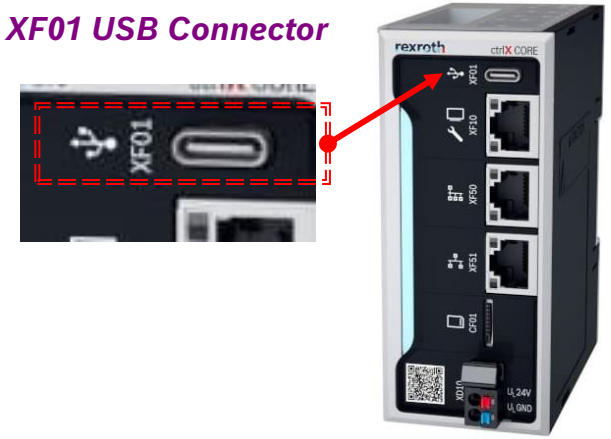
Goals:

In this manual we are going to explain the following sections:

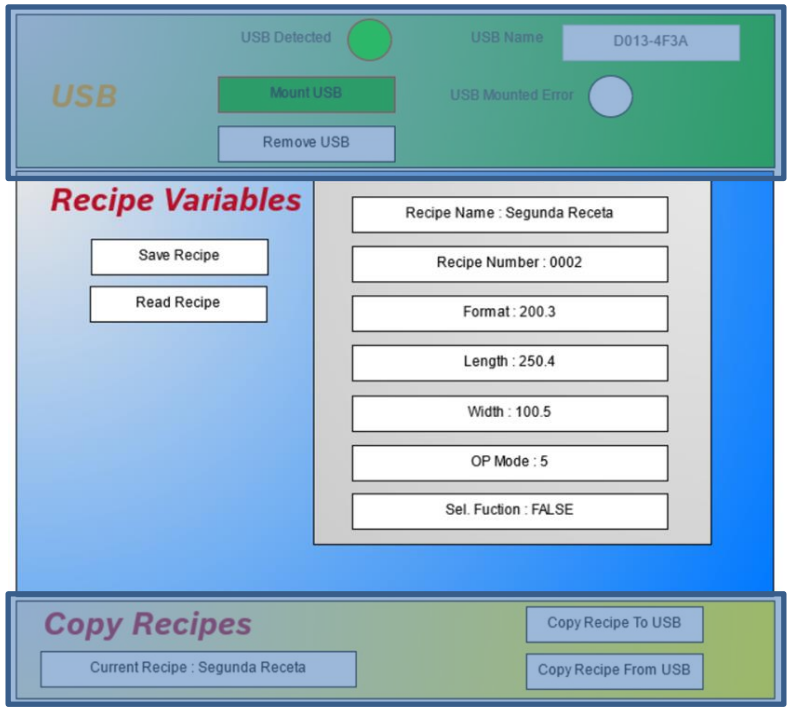
- ***1º Management of recipes from Web visu and PLC program with the use of the "Recipe Management" library***
- ***2º Control of USB from PLC and activate with the use of Nodered***
- ***3º Copy files from PLC to USB of the ctrlX Core X3 (XF01 Connector) and vice versa***

ctrlX

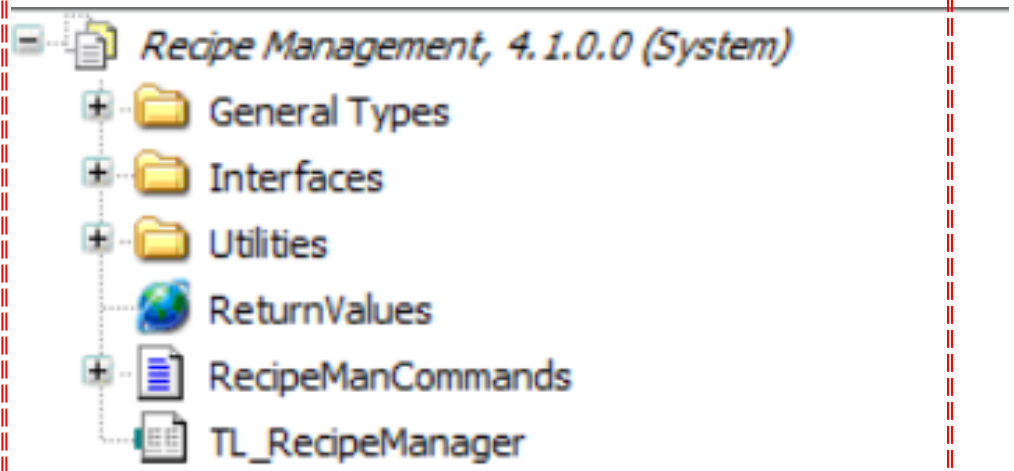
XF01 USB Connector



Web Visu



Library for use of the modules from the PLC



rexroth
A Bosch Company

What is a recipe?

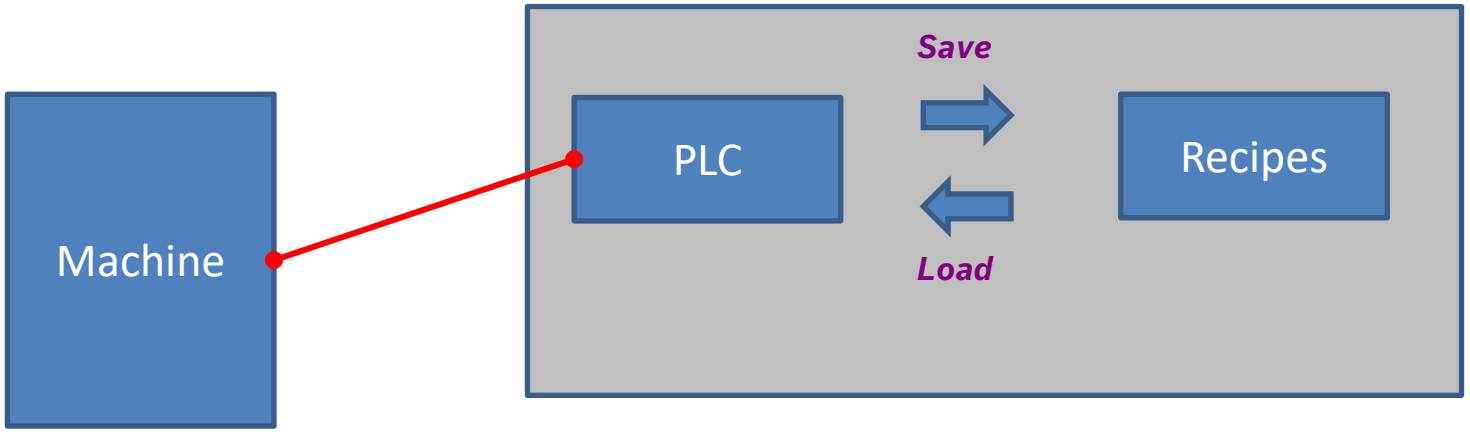
- *In general, the first thing that comes to mind, without doubting it, will be to associate the word, Recipe, with some food, dessert or similar, in other cases, we will also relate it to a medical prescription that has been imposed on us and that will force us to take a medication to counteract some discomfort we have*
- *A recipe, as we well think, is nothing more than a set of ingredients that together will end up becoming something, if it is cook, possibly very appetizing, although sometimes it can go wrong if we do it not well.*
- *For example:*

  900 grams of whole milk
  150 grams of white sugar
  6 Egg yolks
  50 grams of cornflour
  200 grams of milk chocolate
  100 grams of whipping cream
  40 rectangular biscuits

These ingredients are still the vital "elements" of a Recipe and together they will form a chocolate cake

- *In our case, speaking of programming, the Recipe becomes an essential element when in a machine you can use different sets of elements, formats, or other situations that if it weren't for the recipes would become a real madness.*

The PLC controls the functionalities of the machine, and the recipes can be used to define the overall behavior of the machine, depending on the part to be manufactured.



rexroth
A Bosch Company

- ***Without recipes, we would have a machine in which we would have to modify all the configuration values by hand, every time we wanted to change, for example, the cutting size of a material.***

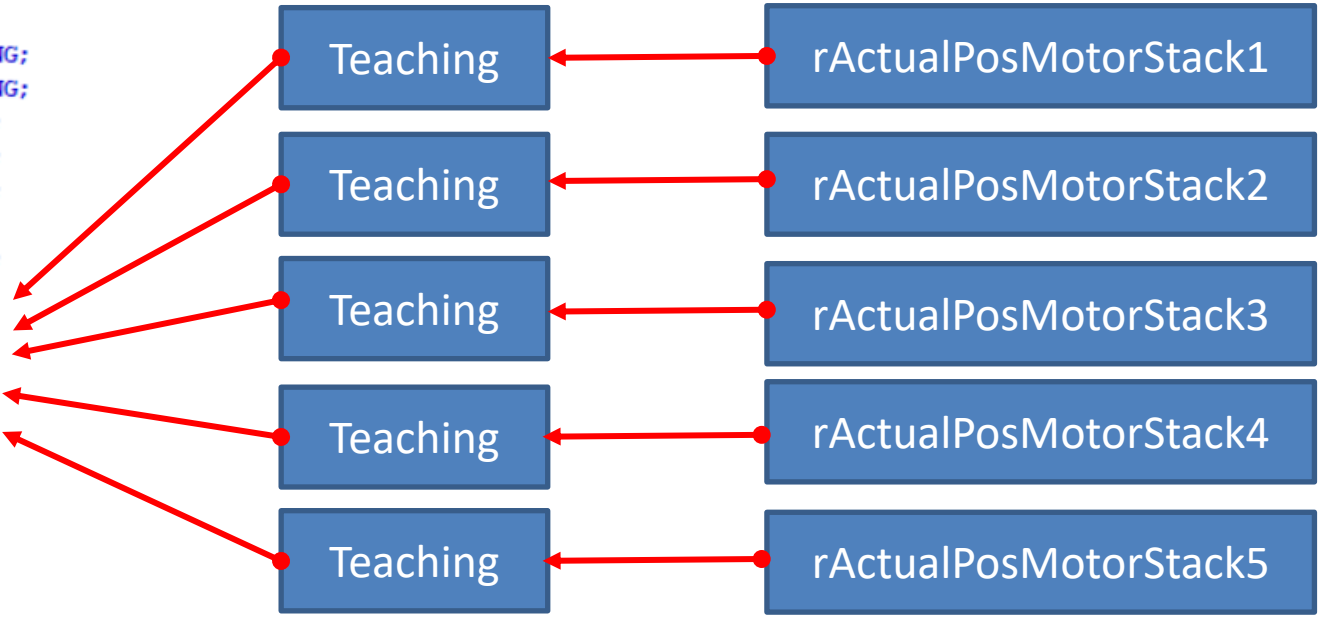
```
STRUCT
strRecipeName    : STRING;
strRecipeNum     : STRING;
rFormat          : REAL;
rlength          : REAL;
rWidth           : REAL;
iOpeMode         : INT;
bSelFunc         : BOOL;
```

Without recipes, we would have to modify all the parameters that affect the part we are going to manufacture. In the example there are only 7 elements. In general, there are usually many more.

- ***Even with recipes, this first time, it should be done anyway or use some "Teaching" system to "extract", for example the current position values of some engines, if they are already in their correct position.***

In this other example, we have added the variables for the position control of 5 motors and we have used an individual "Teaching" for each of them. Obviously, we could use a general "Teaching" and avoid having many more variables

```
strRecipeName    : STRING;
strRecipeNum     : STRING;
rFormat          : REAL;
rlength          : REAL;
rWidth           : REAL;
iOpeMode         : INT;
bSelFunc         : BOOL;
rPosMotorStack1: REAL;
rPosMotorStack2: REAL;
rPosMotorStack3: REAL;
rPosMotorStack4: REAL;
rPosMotorStack5: REAL;
```



rexroth
A Bosch Company

- ***Let's suppose that now we must make a new part and that this new piece to be cut, to be stacked, to be manufactured, is of other dimensions, which will force us to perform the previous operation again, in this case for this second piece.***

The operator, if we do not have recipes, must save the values in a notebook, currently he could also use a mobile phone photo, if the first of the parts has to be manufactured again at some point. Of course, you will also have to do it with the second one.

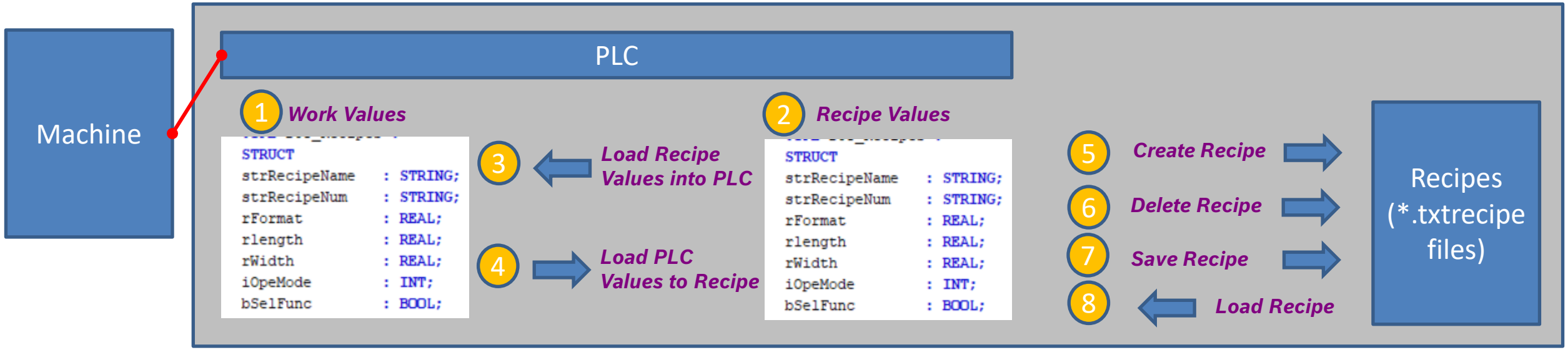
- ***If we have Recipes, the problem mentioned above of the notebook or the photo, somehow ends, since each piece that we must manufacture we can create a specific file for it and that we can recover whenever we want and quickly.***

- ***But what should a prescription have? Obviously, this is debatable and each of us will raise it in our own way and from our point of view:***
  - ***Create Recipe***
  - ***Save Recipe (PLC Values to File)***
  - ***Load Recipe (File Values to PLC)***
  - ***Clear Recipe***
  - ***Sending Recipe Data to Work PLC Values***
  - ***PLC Data Used for Machine Operation***
  - ***PLC data used for the display of recipe values and their manipulation***

As I say, all this is relative, and each programmer will do it in their own way and to their liking

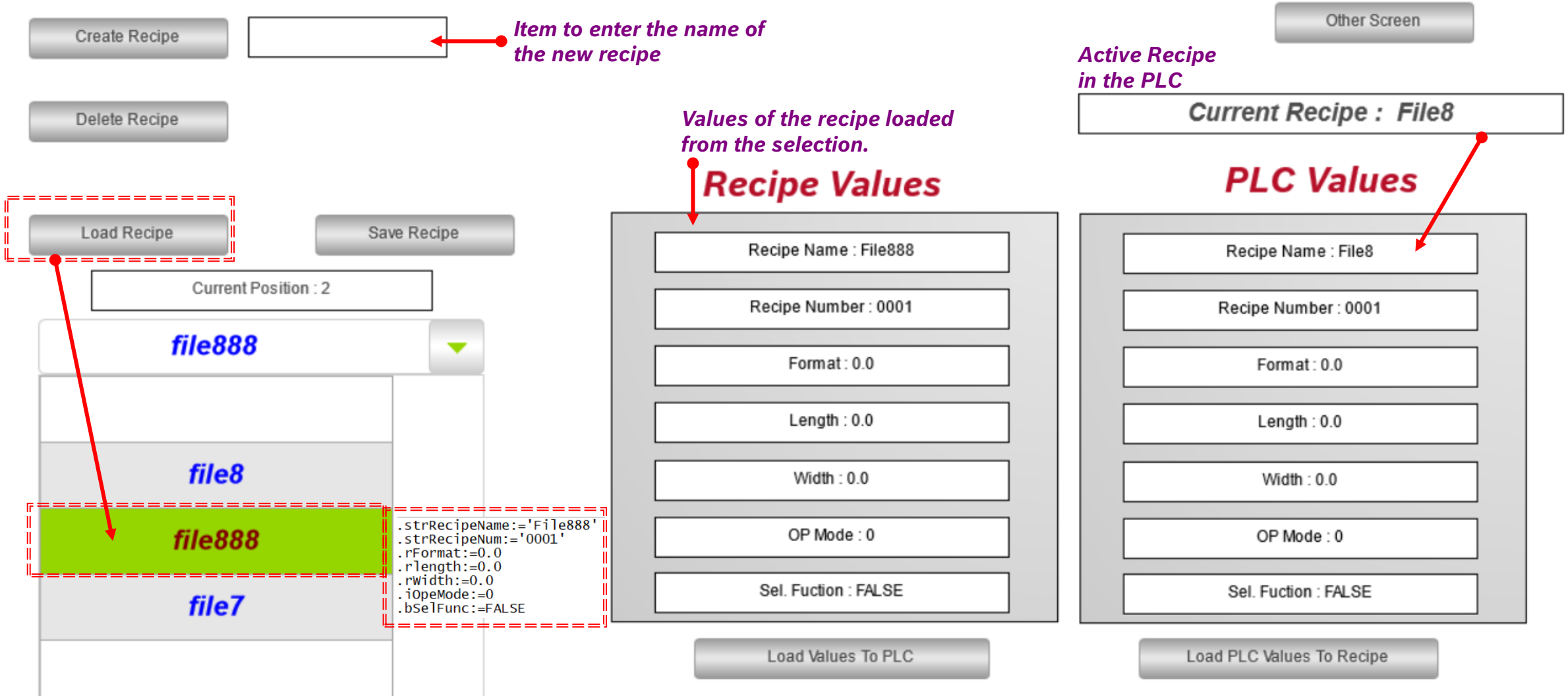
rexroth
A Bosch Company

- ***In the following image you can see a small breakdown of what was discussed:***



1º - Working values: These data are the ones that will be used to make the machine work. They are loaded from the recipe, but always after being validated by the operator.
2º - Current values of the selected recipe. With this option we can work with the recipes without affecting the operation of the machine since the values are visible and manipulable, but they do not affect the work values. Therefore, we will be able to load recipes, manipulate them, delete them, etc., without affecting production.
3rd - The values of the recipe are loaded into the working values. Under no circumstances should it be done indiscriminately, and we should be out of the automatic mode of operation.
4º - Save the recipe with the current values existing in the PLC, writing these in the recipe values and activating the writing of the recipe. Even so, and as a precautionary measure, it should be checked that the recipe to be saved corresponds to the current one, determined by the PLC values, since we could be manipulating another recipe and end up generating a problem. That is why we have point 7 that allows us to save the manipulated recipe.
5º Create Recipe. A new recipe file is created from the values set in the working values.
6º Delete Recipe. Once the recipe is selected, it can be completely deleted from the system.
7º Load Recipe. The values of the selected recipe file are loaded into the PLC variables of the recipe, but not into the working ones, and can be displayed, modified or updated without affecting, as we have said, production.

- ***Although very austere, a screen with these elements could look like this***



rexroth
A Bosch Company

Common data for all modes

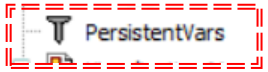
- ***For the following examples, both for the WebVisu and for the one we will use the "Recipe Manager" we are going to use the following operating structures.***

We will generate a main structure (DUT) in which we will place all the variables that we need for the recipe.

```
TYPE DUT_Recipes :
STRUCT
strRecipeName    : STRING;
strRecipeNum     : STRING;
rFormat          : REAL;
rlength          : REAL;
rWidth           : REAL;
iOpeMode         : INT;
bSelFunc         : BOOL;
END_STRUCT
END_TYPE
```

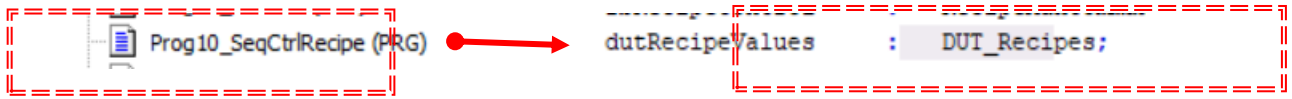
Add the "PersistentVars" object and define the recipe structure for PLC values based on the created DUT.

As they are permanent, they will always keep the last value assigned. In this way, if the Recipe to be used in the next start of the machine is the same, it will not be necessary to load it again. Variables, defined as "Persistent" in other modules or in other variable tables. can be included in this table.



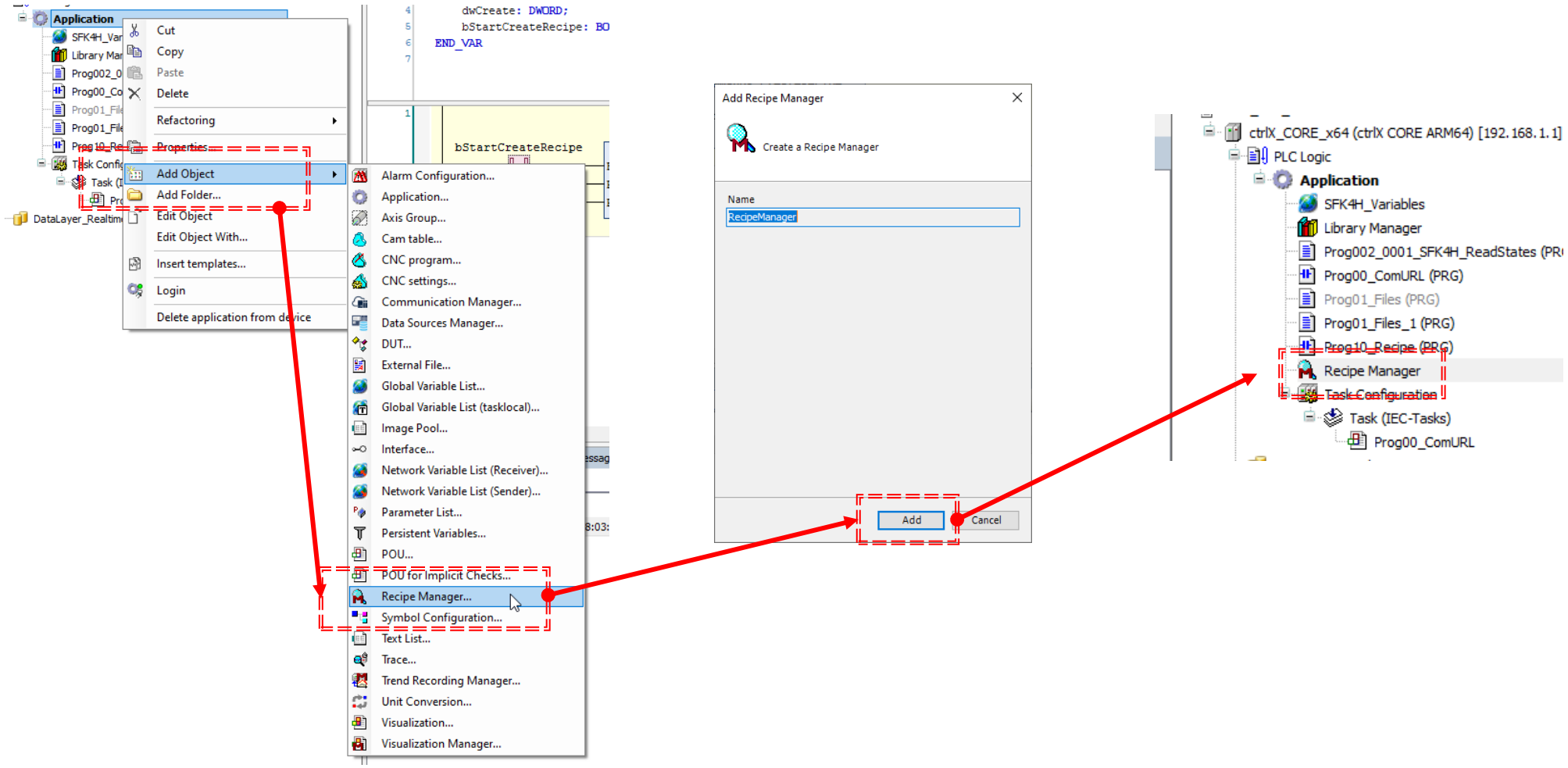

```
VAR_GLOBAL PERSISTENT RETAIN
// Exemple for Recipes Test
// Recipe Definition
  dutRecipePLC: DUT_Recipes;
END_VAR
```

Elsewhere in the program, we add the structure that the recipes will use. In the example, this statement is located in the recipe operation control module.



rexroth
A Bosch Company

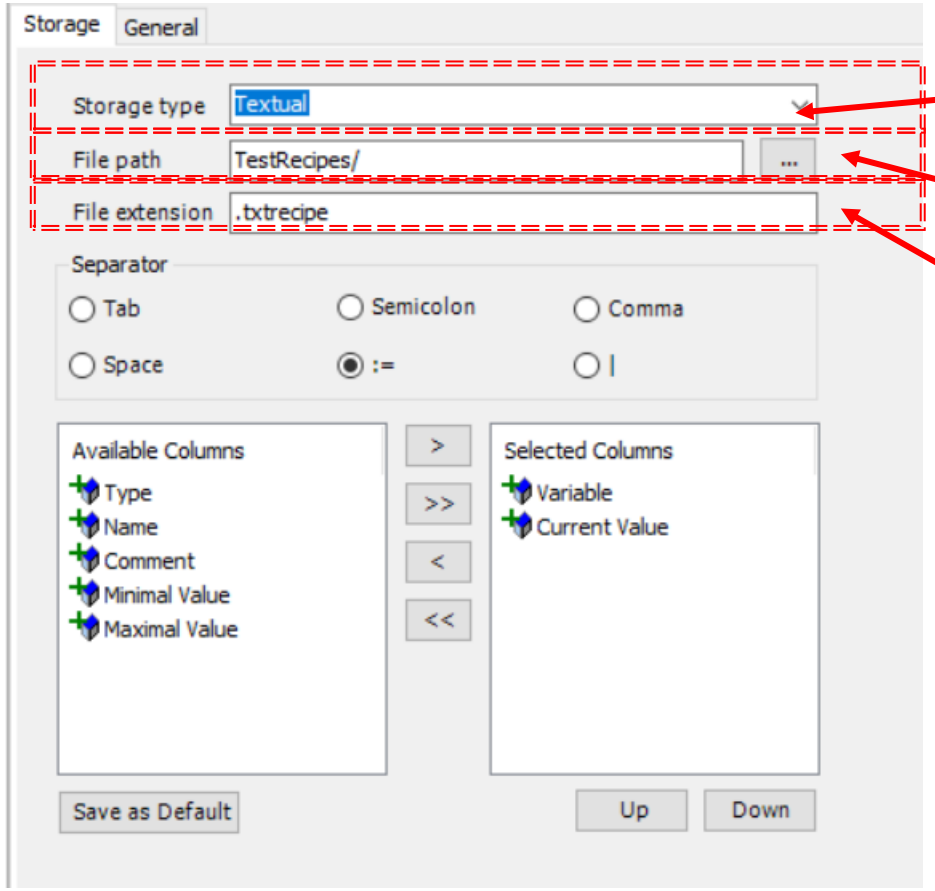
- Next, we will have to insert the "Recipe Manager"”



rexroth
A Bosch Company

- ***Once created, the "Recipe Manager" has two menus of options.***

"Storage":



Safeguard format. Better to leave it in Textual than in binary.

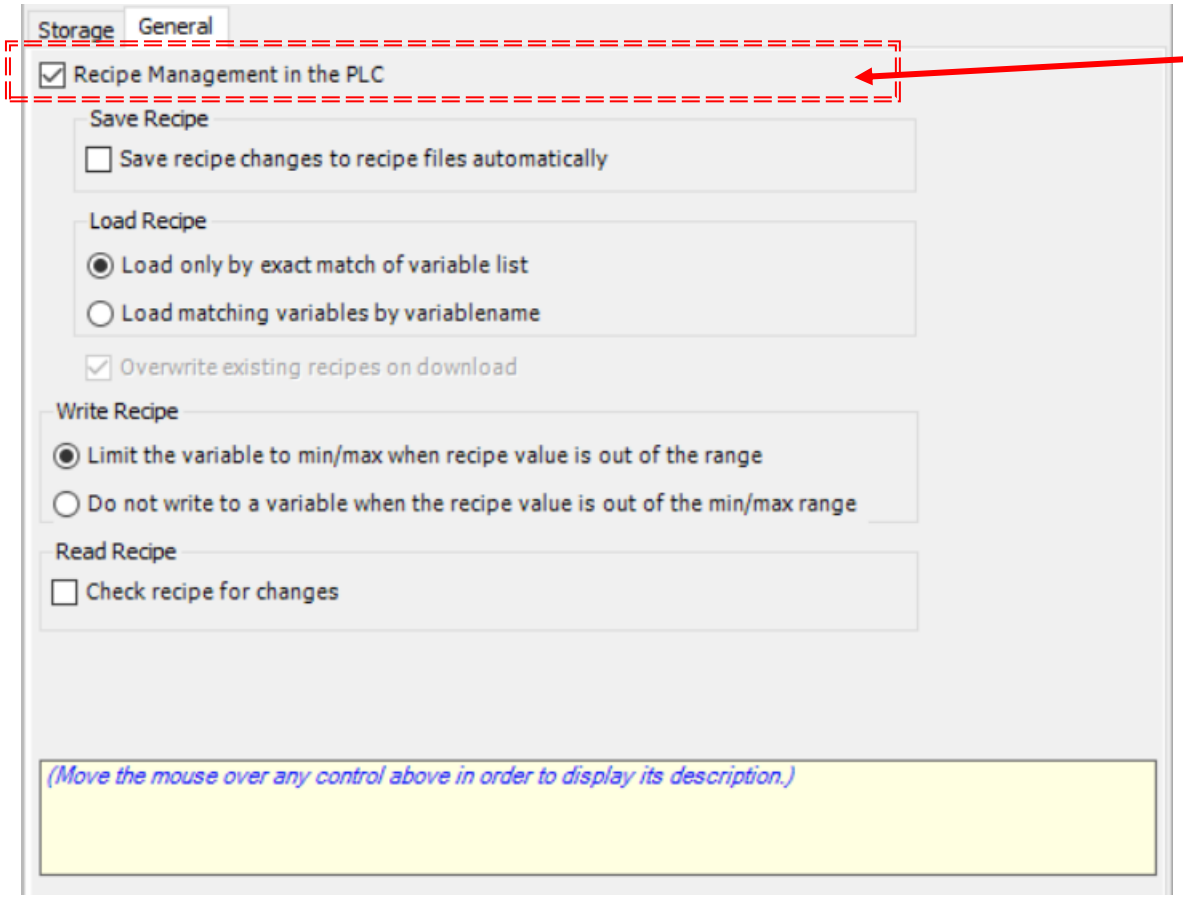
Path where the Recipes will be stored. Useful for not having them in the main folder and being able to access them easily. In the example we are using the auxiliary folder 'TestRecipes/'.

Extension of the recipes that we are not going to modify for the time being.

The rest of the options, for the time being, we are not going to change.

rexroth
A Bosch Company

- *“Recipe Manager" Second menu of "Recipe Manager" "General":*

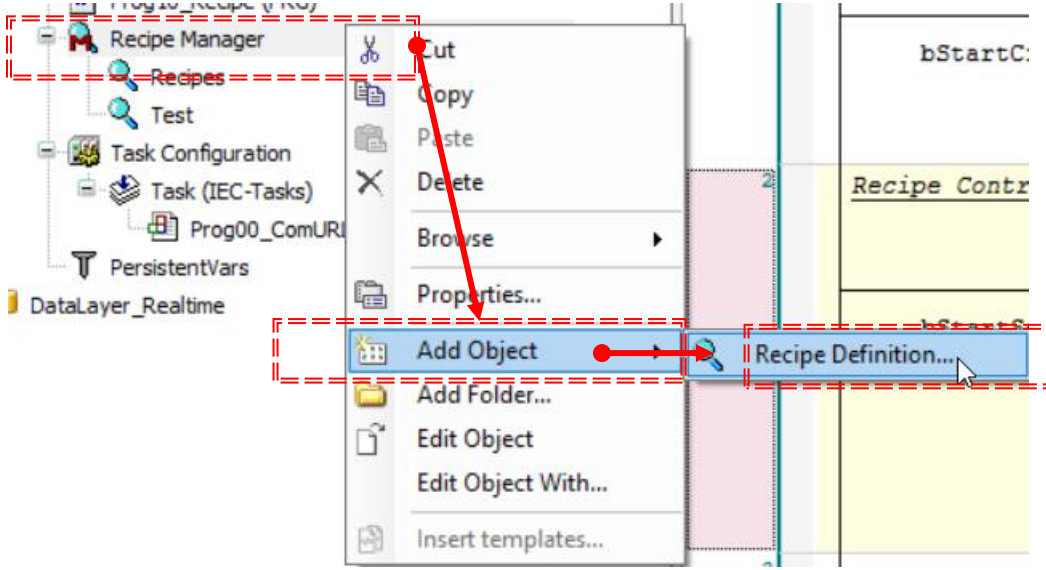


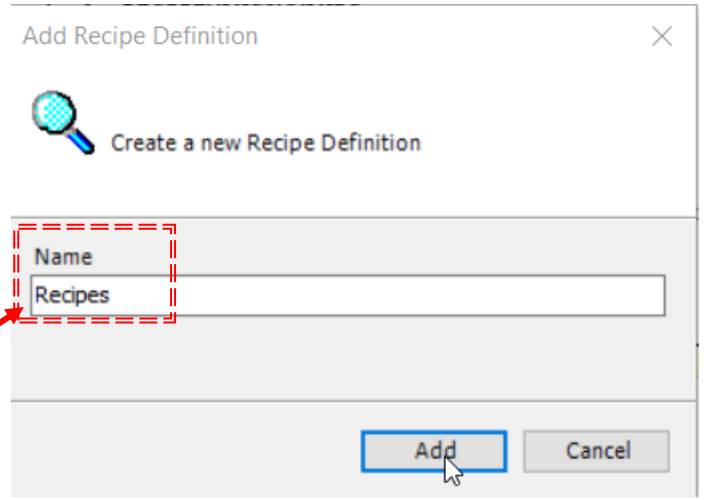
Activate to be able to use the PLC modules

The rest of the options should look as they appear in the image

rexroth
A Bosch Company

- To generate the "Recipes" tables, which can be several, we must generate them from the "Recipe Manager" menu"



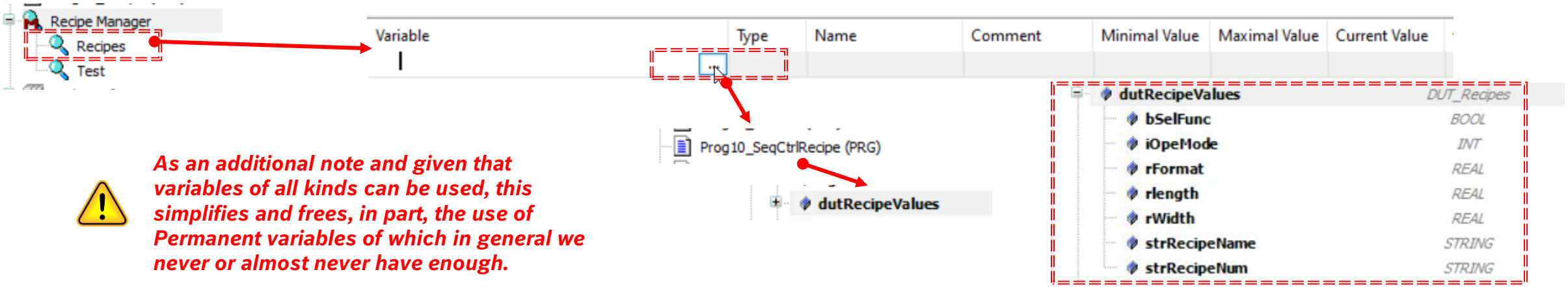


Recipe modules created in the example:
Recipes : Used for the example of recipes.
Test: Other variables.

Another manual details the operation of the "Recipes" at the level of an Indracontrol XM. The overall system is generic for both teams.

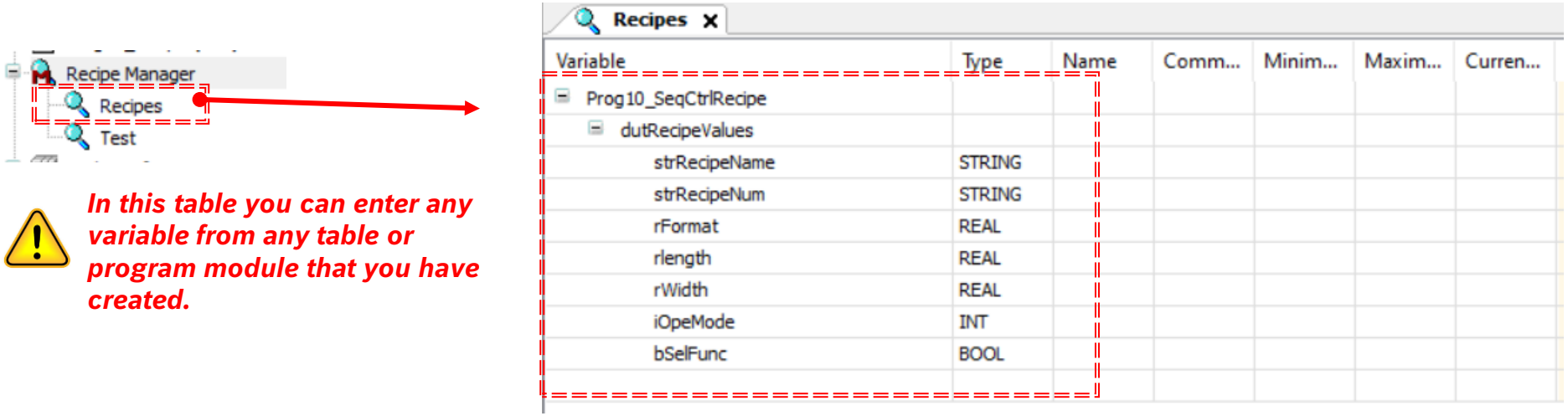
rexroth
A Bosch Company

- ***When we have created the "Recipes" module, we will proceed to insert the variable of the recipe structure that we have defined previously***



As an additional note and given that variables of all kinds can be used, this simplifies and frees, in part, the use of Permanent variables of which in general we never or almost never have enough.

- ***Here we can already see the variables of the structure used in the Recipes table***



In this table you can enter any variable from any table or program module that you have created.

rexroth
A Bosch Company

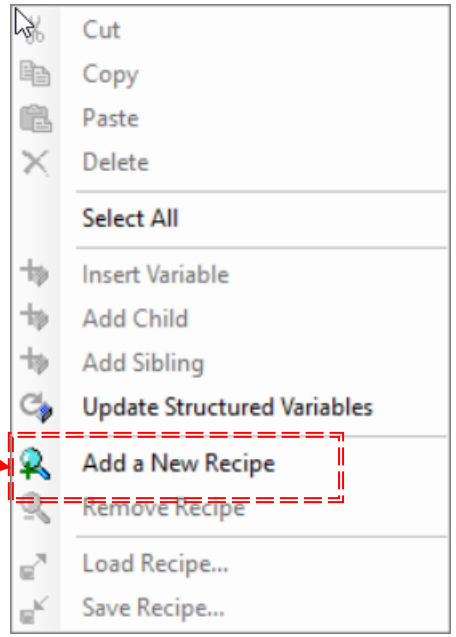
- ***Finally, and before starting, we are going to insert an Initial Recipe integrated into the system and that will allow us to access it from WebVisu and from the program controlled by the PLC***

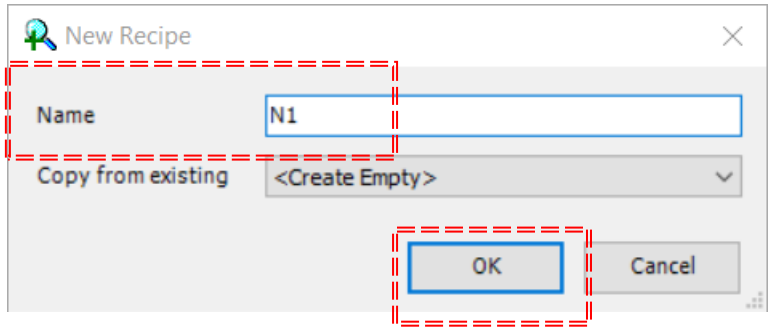
Access the recipe structure "Recipes"

Recipes

Variable	Type	Name	Comm...	Minim...	Maxim...	Curren...
Prog10_SeqCtrlRecipe						
dutRecipeValues						
strRecipeName	STRING					
strRecipeNum	STRING					
rFormat	REAL					
rlength	REAL					
rWidth	REAL					
iOpeMode	INT					
bSelFunc	BOOL					

From any point, activate the options menu and choose "Add a New Recipe"



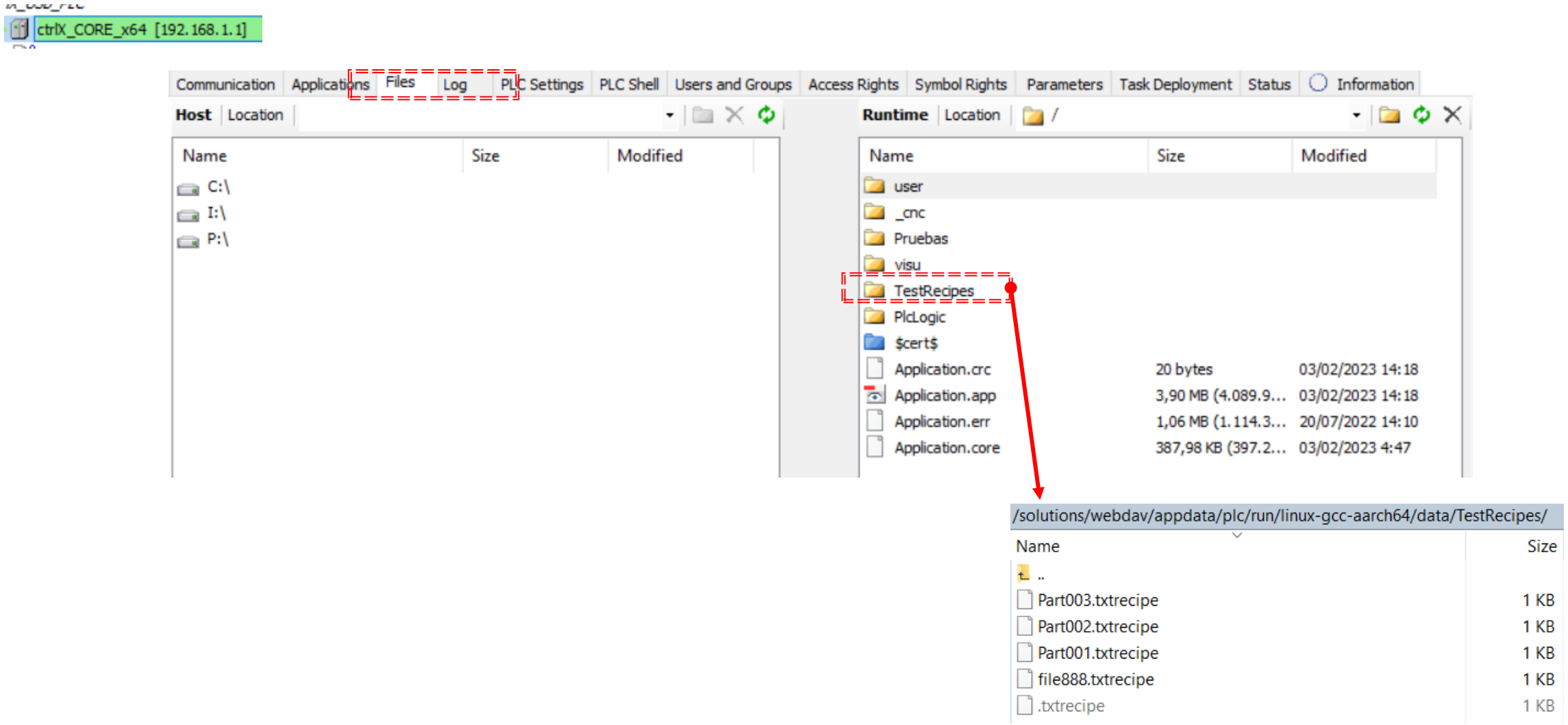


After activating "Ok" we get the new recipe

Variable	Type	Name	Comment	Minimal Value	Maximal Value	Current Value	N1
stRecipe							
strRecipeName	STRING						
strRecipeNum	STRING						
rFormat	REAL						
rlength	REAL						
rWidth	REAL						
iOpeMode	INT						
bSelFunc	BOOL						

rexroth
A Bosch Company

- The created files can be viewed from the ctrlX side of the "Files" option



rexroth
A Bosch Company

- We can also visualize them, for example, with a program such as WinSCP FTP Client

WinSCP FTP Client

Configuration options for communicating with the ctrlX Core

Session

File protocol:	Encryption:
WebDAV	TLS/SSL Implicit encryption

Host name:	Port number:
192.168.1.1	443

User name:	Password:
boschrexroth	••••••••••••••••

Edit Advanced...

/solutions/webdav/appdata/plc/run/linux-gcc-aarch64/data/TestRecipes/

Name	Size	Changed	Rights
..		03/02/2023 15:36:01	
Part003.txtrecipe	1 KB	03/02/2023 15:36:01	
Part002.txtrecipe	1 KB	03/02/2023 15:35:56	
Part001.txtrecipe	1 KB	03/02/2023 15:35:49	
file888.txtrecipe	1 KB	03/02/2023 15:16:18	
.txtrecipe	1 KB	03/02/2023 12:48:53	

```
Prog10_SeqCtrlRecipe.dutRecipeValues.strRecipeName:='File888'
Prog10_SeqCtrlRecipe.dutRecipeValues.strRecipeNum:='0001'
Prog10_SeqCtrlRecipe.dutRecipeValues.rFormat:=0.0
Prog10_SeqCtrlRecipe.dutRecipeValues.rlength:=0.0
Prog10_SeqCtrlRecipe.dutRecipeValues.rWidth:=0.0
Prog10_SeqCtrlRecipe.dutRecipeValues.iOpeMode:=0
Prog10_SeqCtrlRecipe.dutRecipeValues.bSelFunc:=FALSE
```

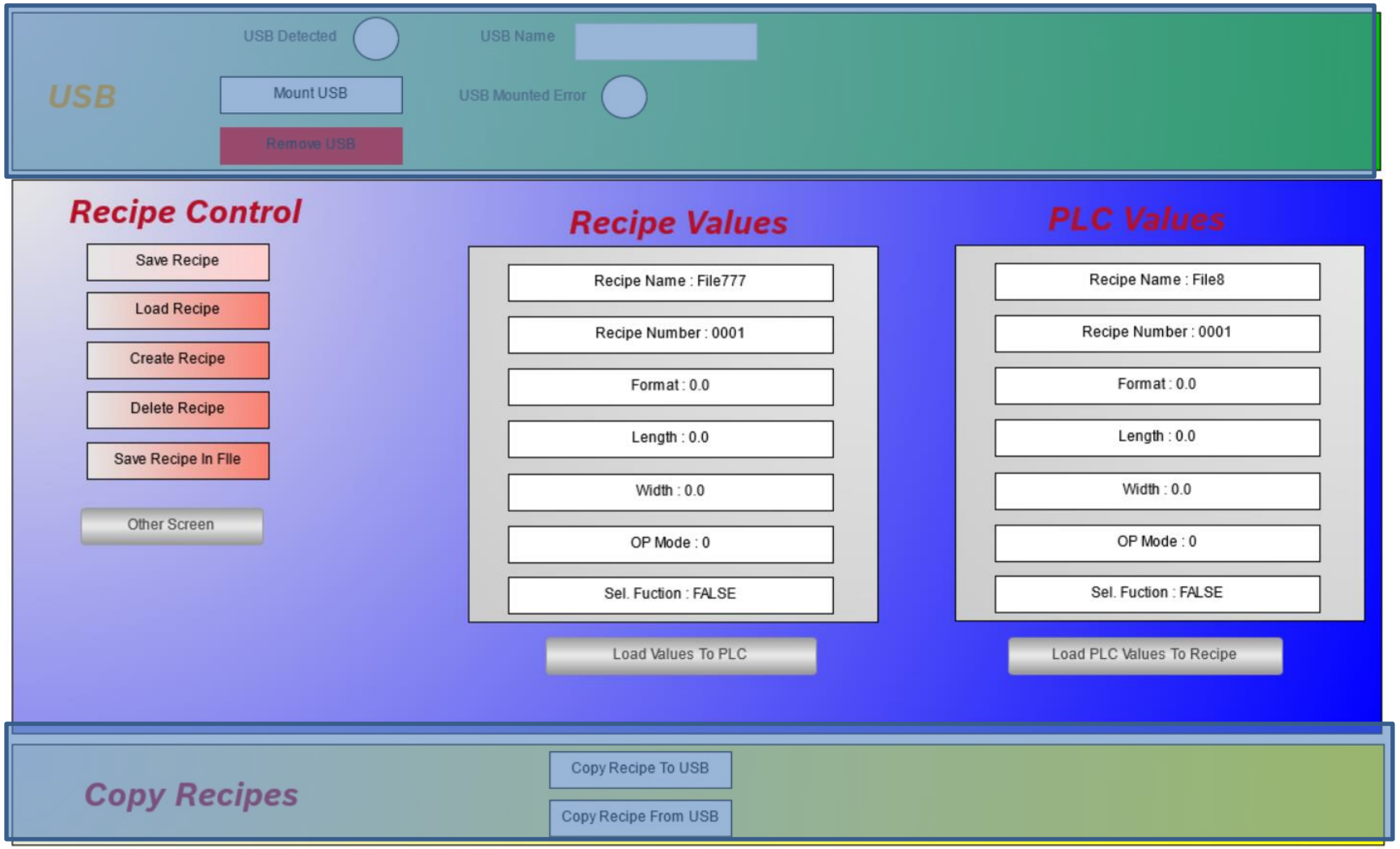
dutRecipeValues	DUT_Recipes	
strRecipeName	STRING	'File888'
strRecipeNum	STRING	'0001'
rFormat	REAL	0
rlength	REAL	0
rWidth	REAL	0
iOpeMode	INT	16#0000
bSelFunc	BOOL	FALSE

From here, you can view and even modify the files generated by the recipe system

rexroth
A Bosch Company

Recipe Control Using WebVisu

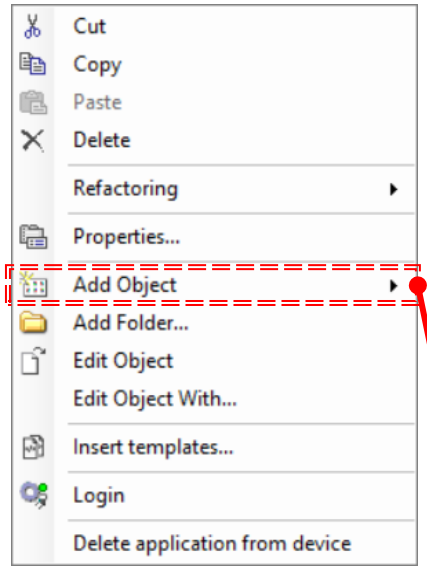
- The first thing we're going to look at is how to use the WebVisu option to manage recipes. From this part, it is easy to control the options for creating, loading, saving and deleting from the elements of the control screen

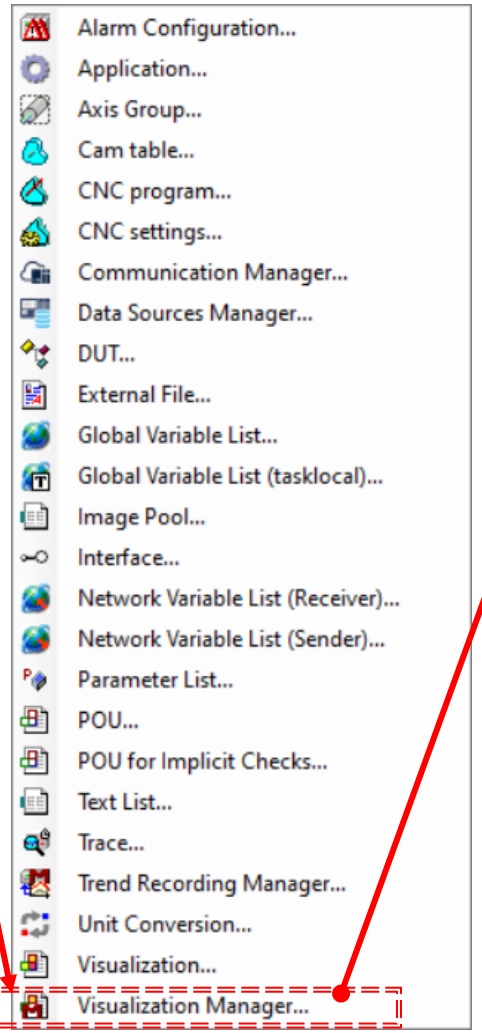


rexroth
A Bosch Company

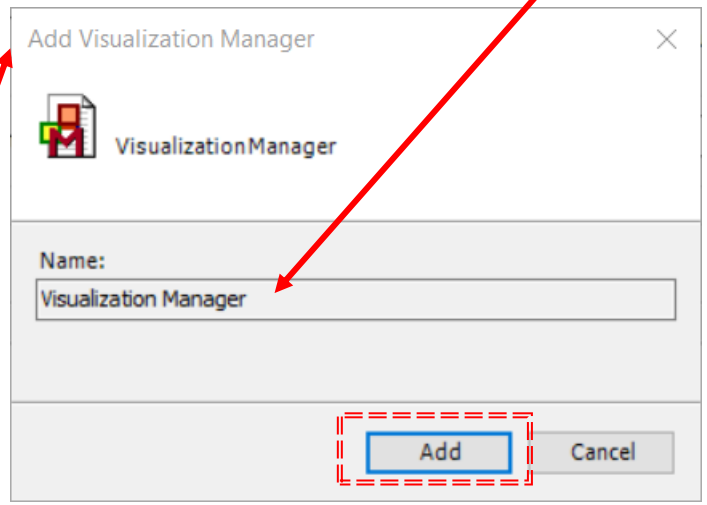
- The first step will be to include the "Visualization Manager" object

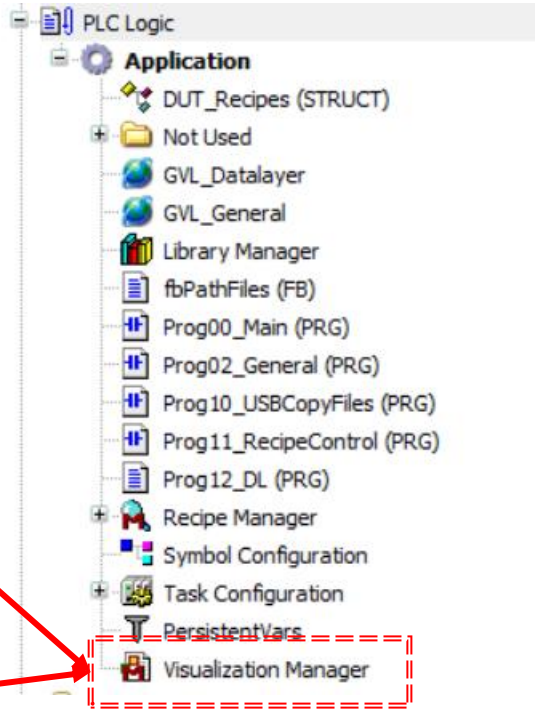
This option will also be useful for the use of the "Recipe Manager"”





We can modify the name or leave it as is.

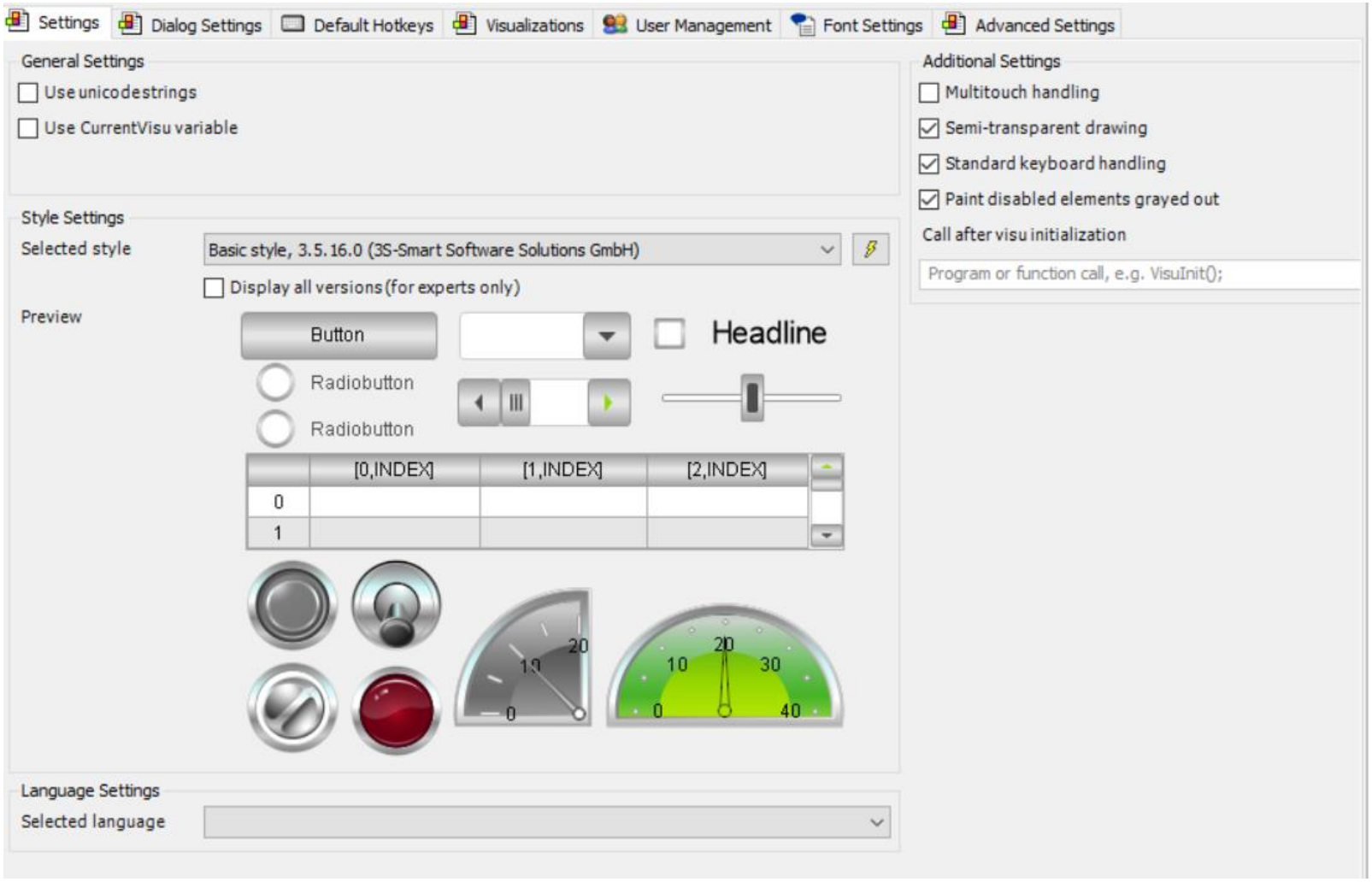




The new object should appear in the PLC element structure.

rexroth
A Bosch Company

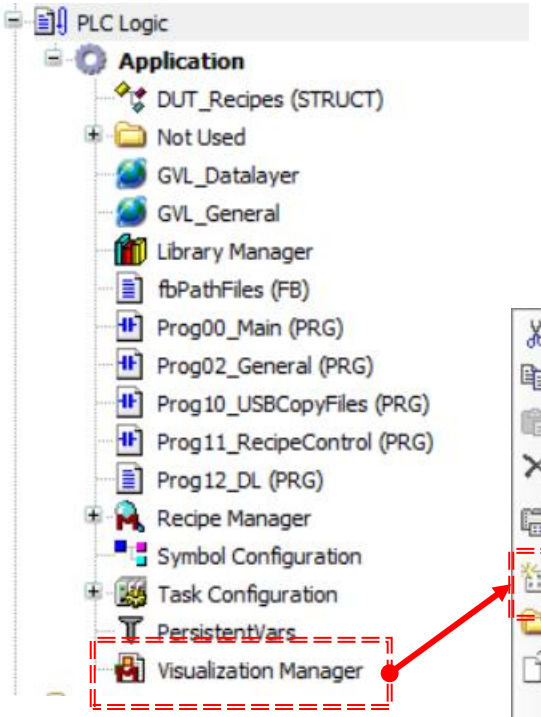
- The "Visualization Manager" has several optional screens to modify its functionalities.

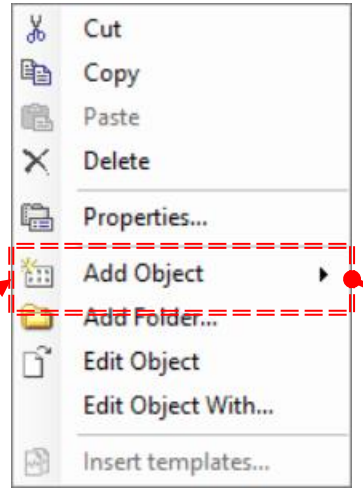


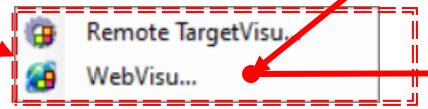
We are not going to change anything at the moment.

rexroth
A Bosch Company

- From "Visualization Manager" we can choose two options (using the drop-down menu)

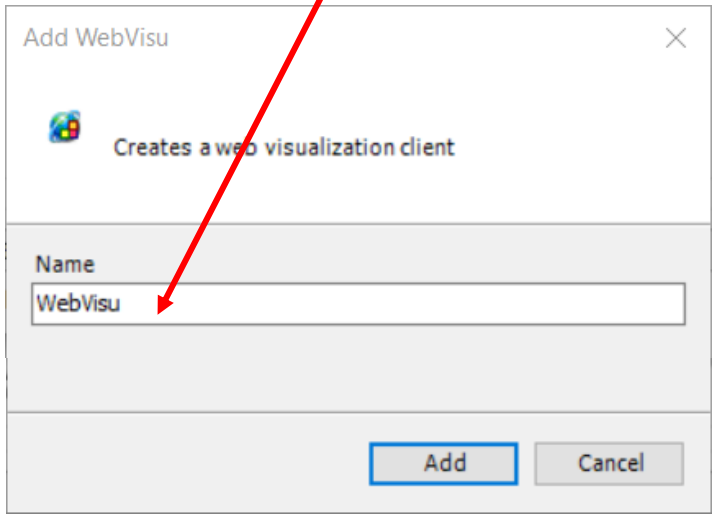






In this case, we will choose the "WebVisu" option.

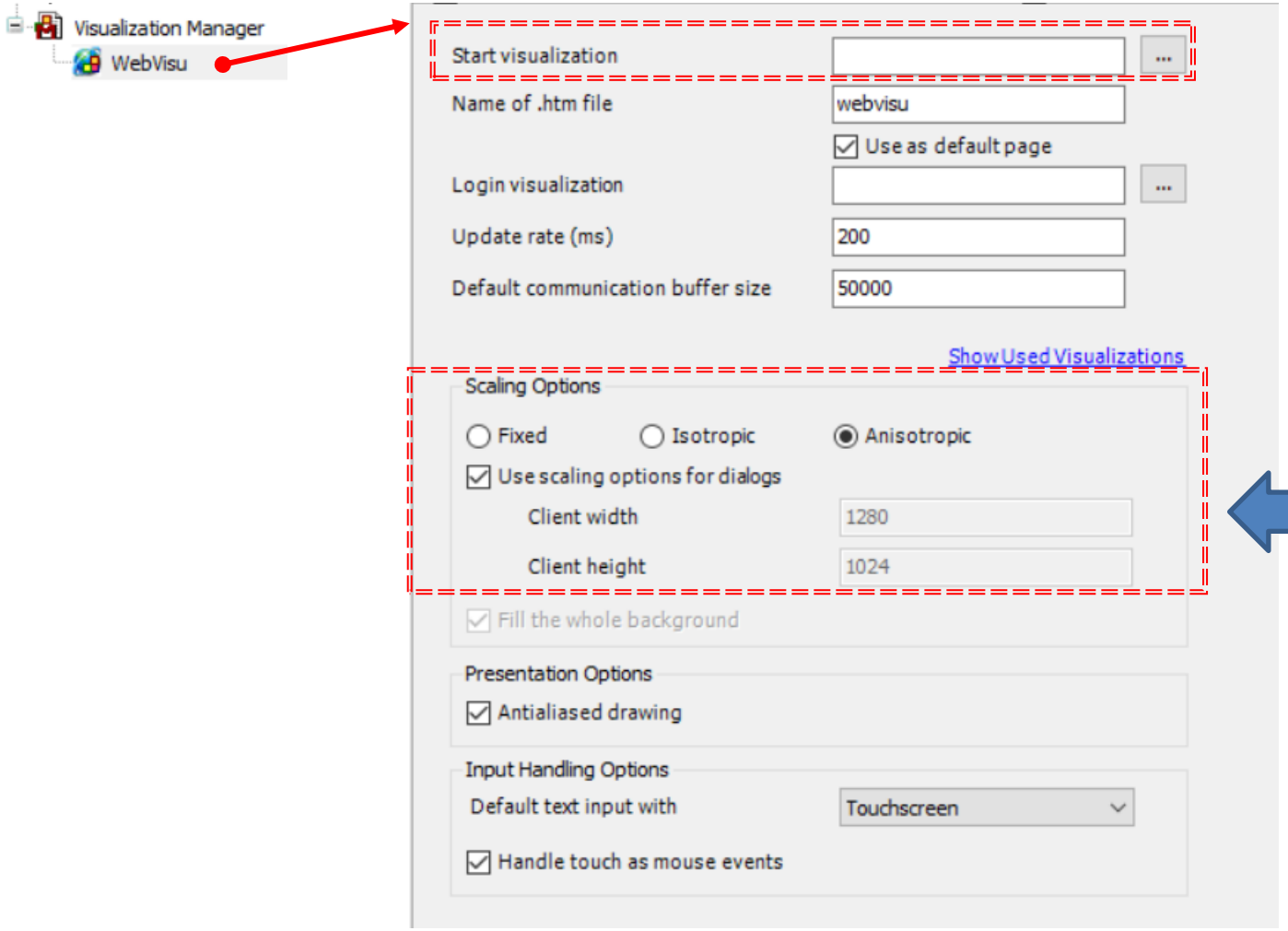
As in the previous case, we can modify the name or leave it as it is.



This option will also be useful for the use of the "Recipe Manager""

rexroth
A Bosch Company

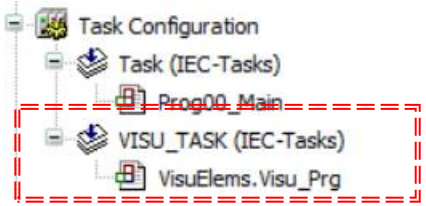
- In the image we can see the selectable options of the added "WebVisu":



We're not going to change anything at the moment Although we will have to add the screen with which we want to start the "WebVisu", but since we do not have it generated, we will do it later.

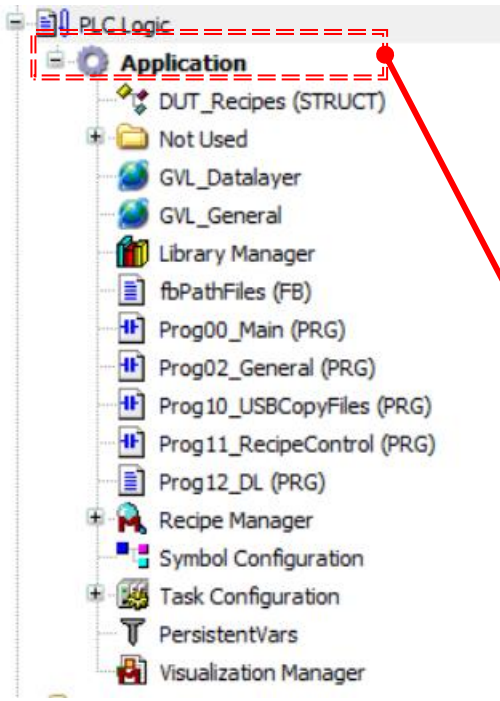
Size of the screen to be used

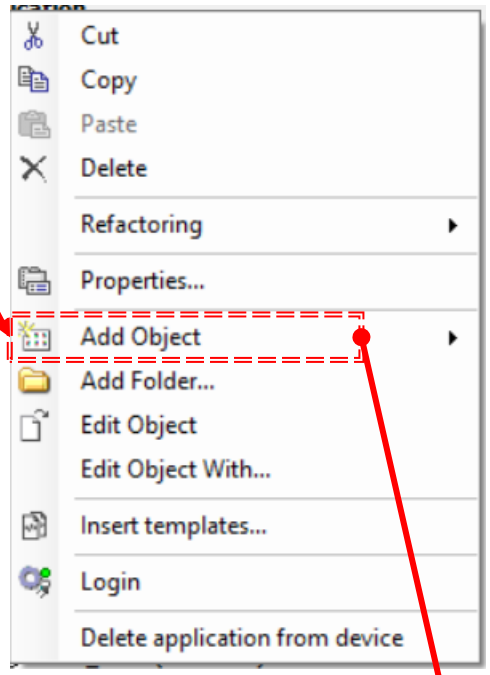
The creation of the "WebVisu" elements has also created a new task for screen control.

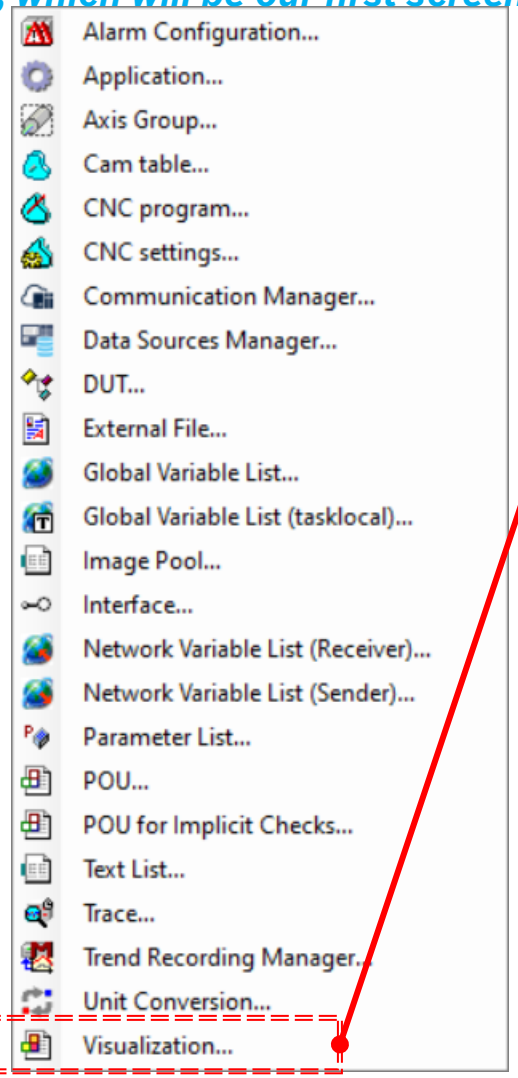


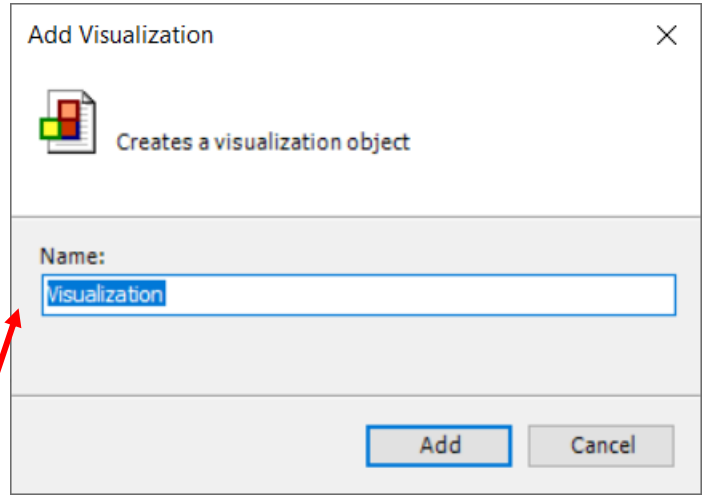
rexroth
A Bosch Company

- Next, we'll add another object, in this case "Visualization", which will be our first screen:







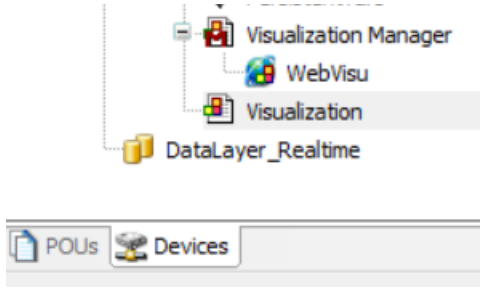


As in the previous cases, we can modify the name or leave it as it is.

rexroth
A Bosch Company

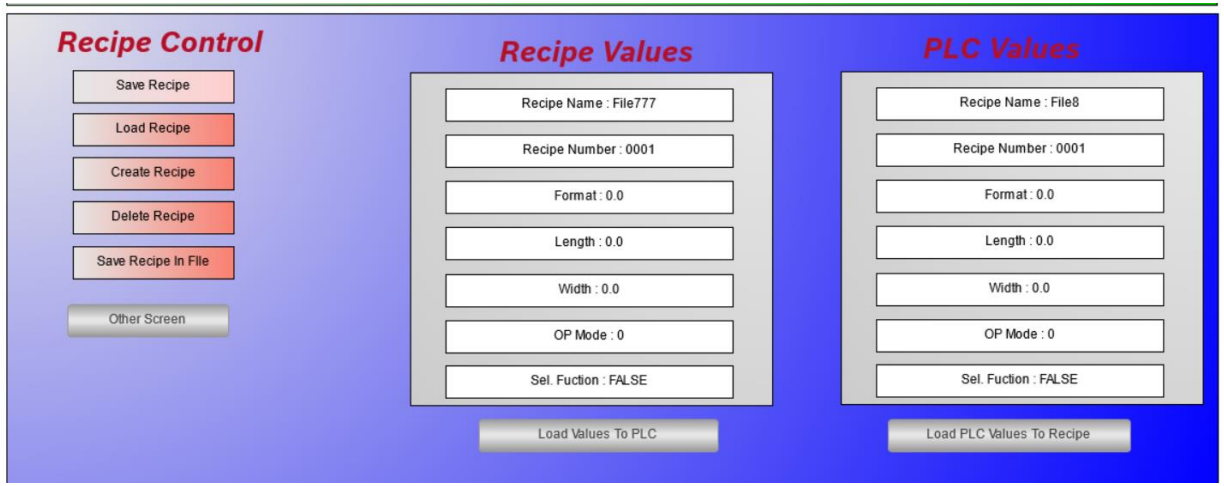
- The "Visualization" element should appear in the "Visualization Manager" area, however on some occasions the visualizations appear in the "POUs" part as can be seen in the images. The operation is the same, however they are in different places.

Visualization at the same point as the "Visualization Manager"



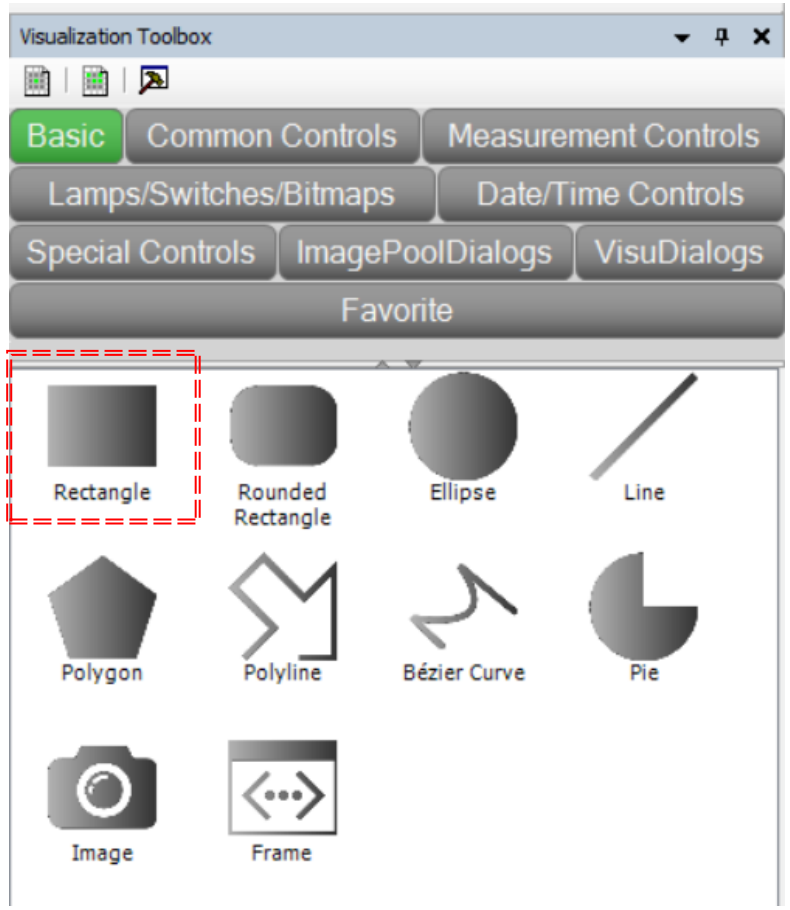
Visualization in the "POUs" section

- On the screen, as we have already mentioned, we will have the buttons to control the variables that we have already described in the common data section.

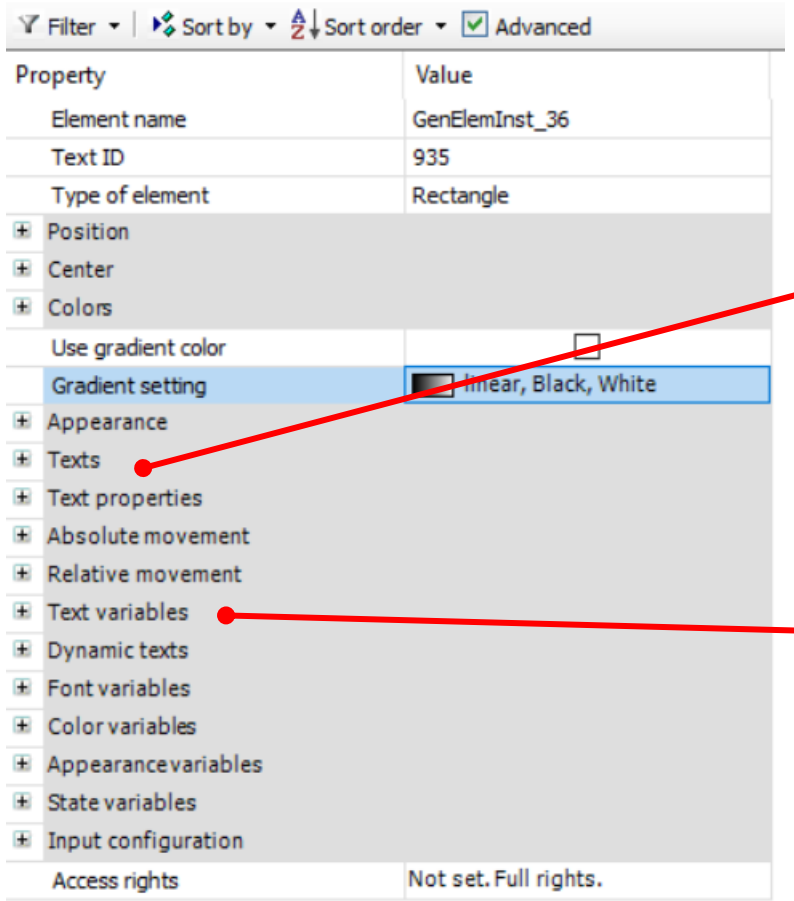


rexroth
A Bosch Company

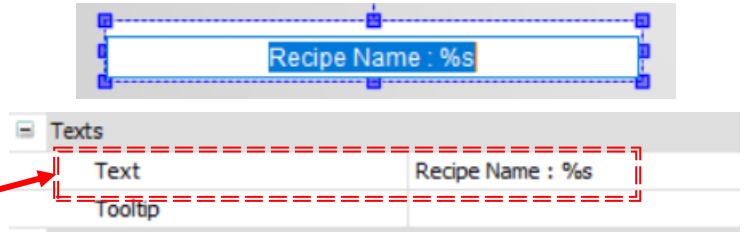
- Defining Elements for Variables:



If we drag the "rectangle" towards the visualization, the properties of the object should appear.

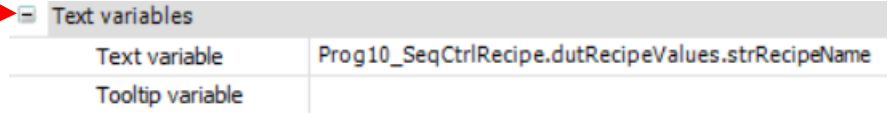


We can directly type the name of the element. If we want, which is the case that it is a variable text, we must place, if we want, first we have to reference and secondly the %s statement that converts the text into a variable argument



The following page details the different allocation options with the % indicator

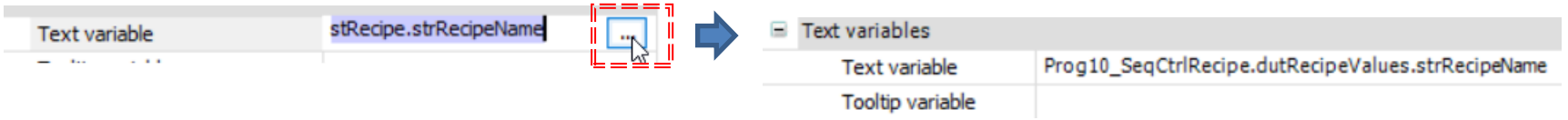
From this other section, you can select the variable that you want to use to read or write.



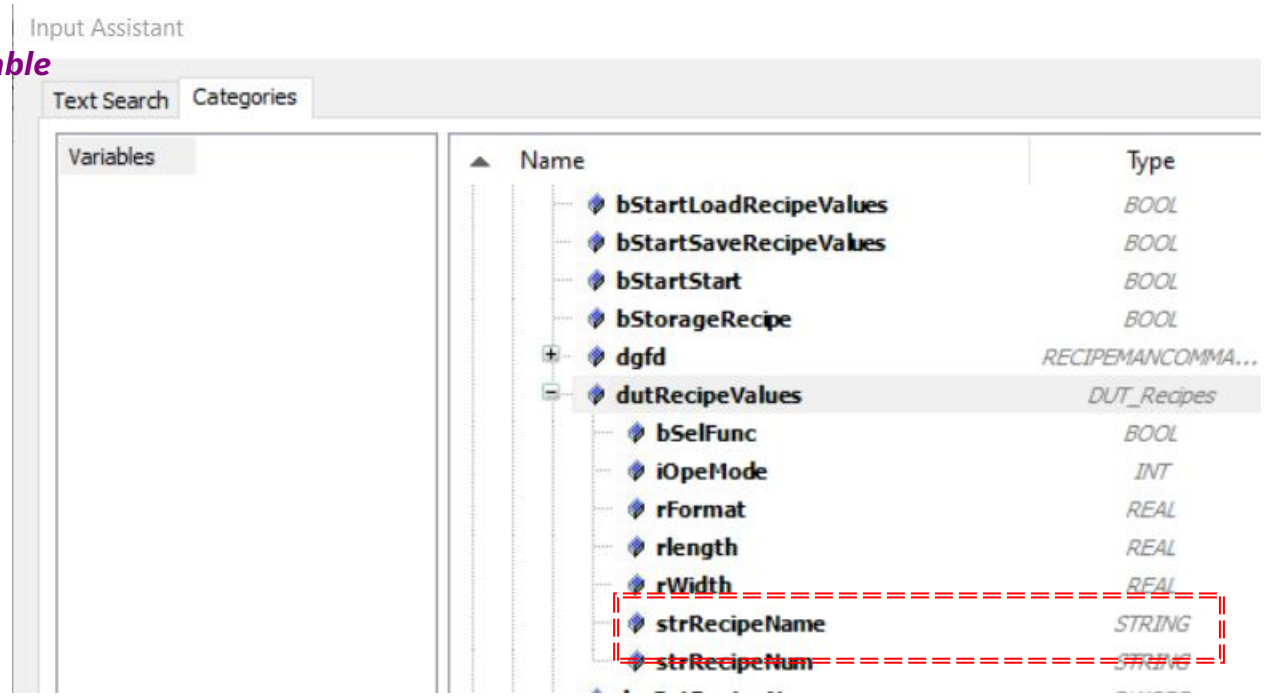
rexroth
A Bosch Company

- The variable used can be found as follows (remember that it was part of the structure that we had already defined for the recipe in the previous example):

When you select the option, "..." with which we can access the system of variables

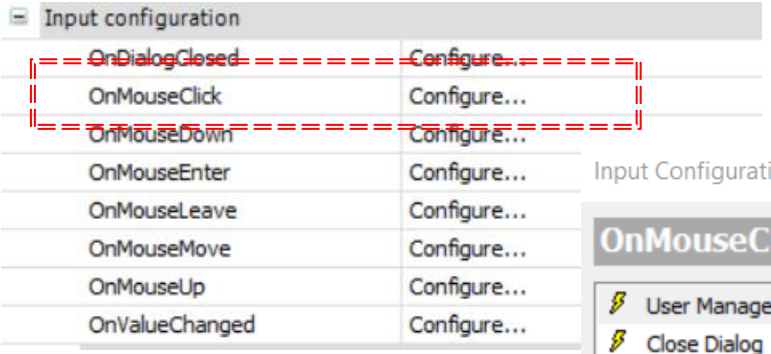


Choose the variable and you'll have it available to use:



rexroth
A Bosch Company

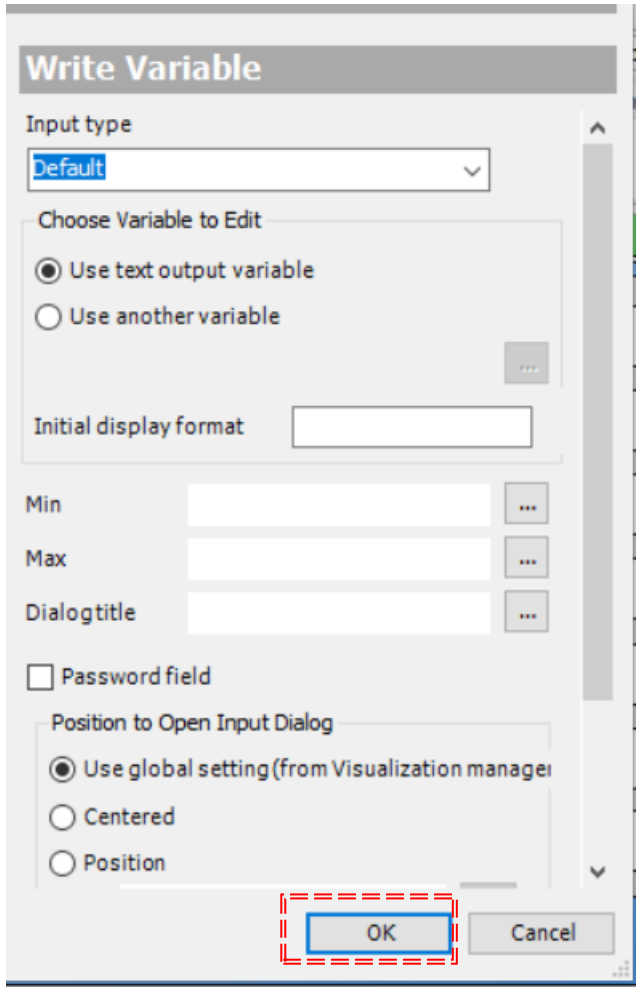
- We can also define the way the variable acts and since in this case what we are going to do is write the value of the variable on the screen, we will have to modify the following section "Input Configuration" (without forgetting that this is just an example)



The next drop-down menu will open, in which we will select "Write Variable"

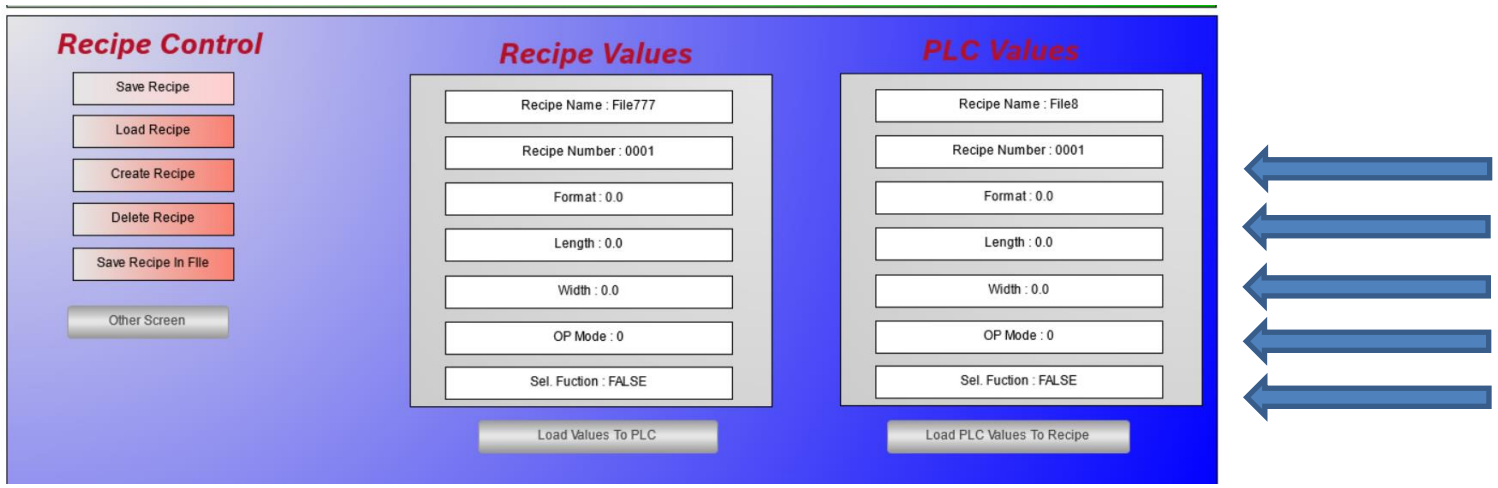
rexroth
A Bosch Company

- And we'll leave the default values

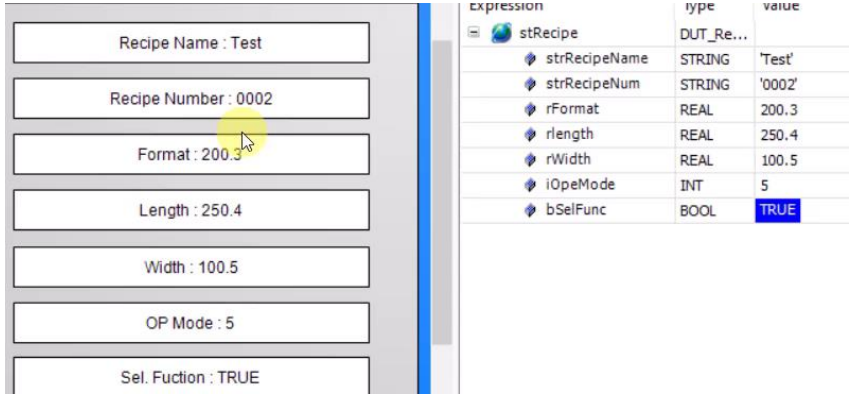


Validate the changes with "OK

The operation performed with the "Recipe Name" variable must be performed with the rest of the recipe variables used

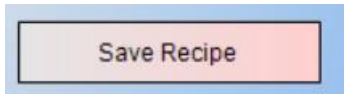


Short video in which we can see how the screen variables are written in the PLC



rexroth
A Bosch Company

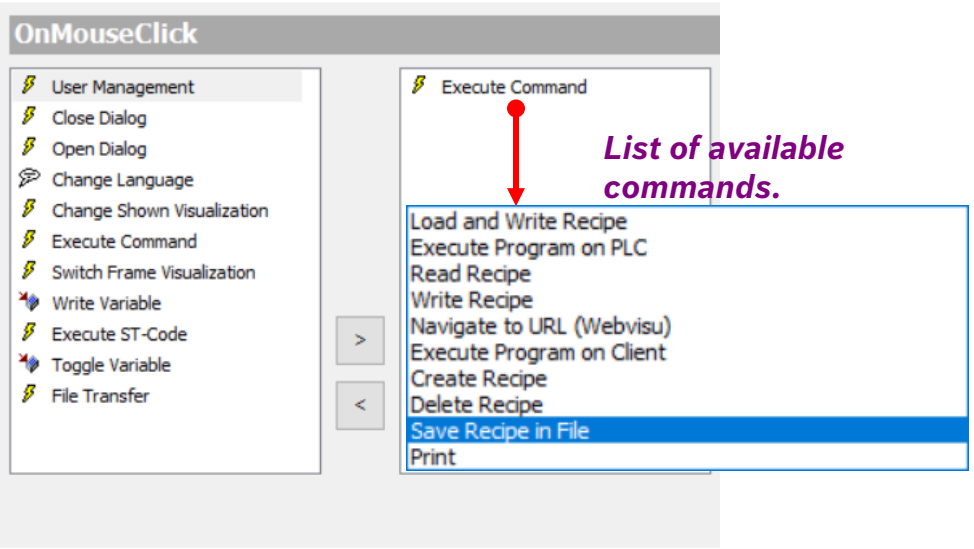
- The next step is to get the system from the screen to allow the loading and safeguarding of the variables in the recipes, for this we are going to incorporate two new elements of type "Rectangle" and that we will call "Save Recipe" and "Load Recipe"



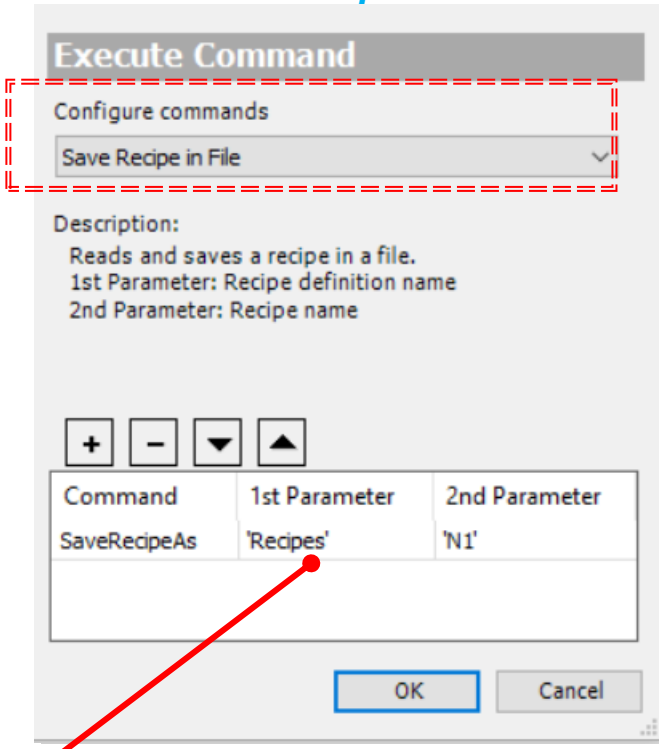
Texts	
Text	Save Recipe
Tooltip	

Input configuration	
OnDialogClosed	Configure...
OnMouseClick	Configure...

Input Configuration



List of available commands.



The command in question has two parameters available. The first is the "Recipe Definition Name" and the second is the Recipe we had created before "N1"

Configured Command:

OnMouseClick	Configure...
Execute Command	SaveRecipeAs, 'Recipes', 'N1'

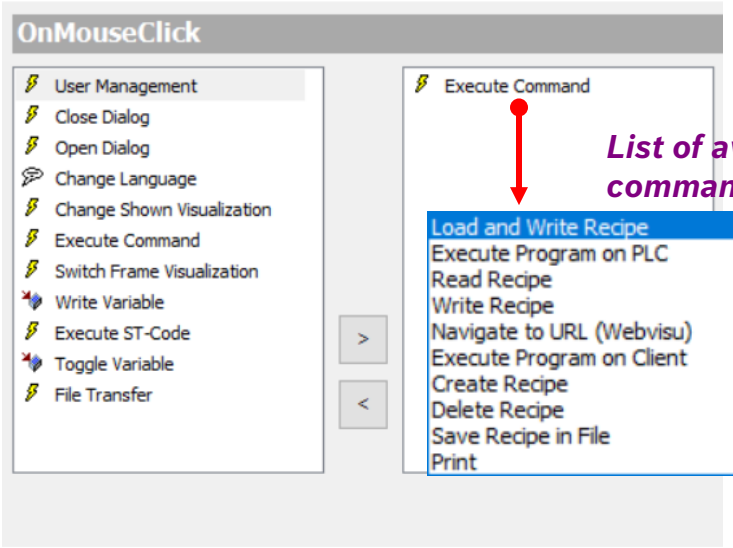
rexroth
A Bosch Company

- The "Load Recipe" should be configured in the same way:

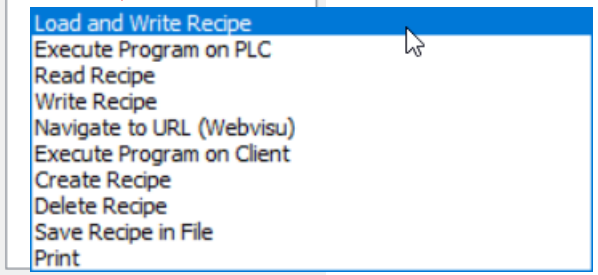
Texts	
Text	Load Recipe
Tooltip	

Input configuration	
~~OnDialogClosed~~	~~Configure...~~
OnMouseClick	Configure...

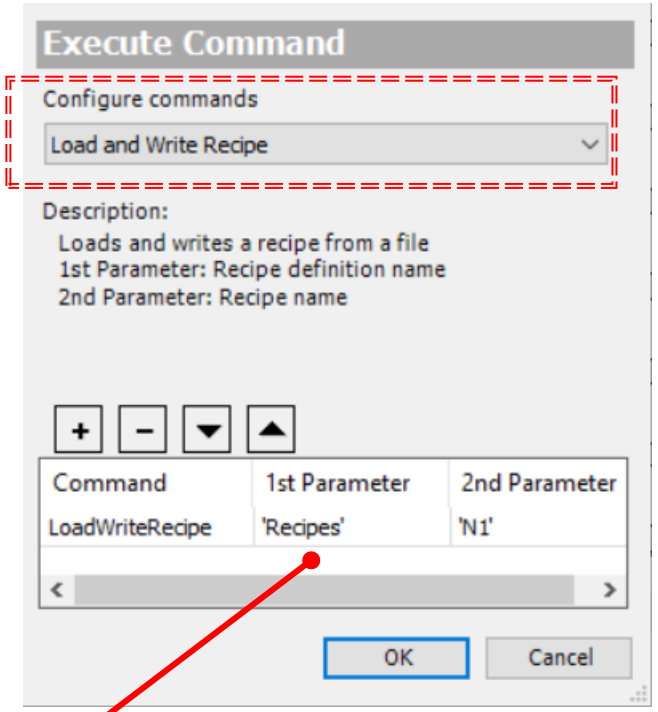
Input Configuration



List of available commands.



The rest of the buttons must be created in the same way



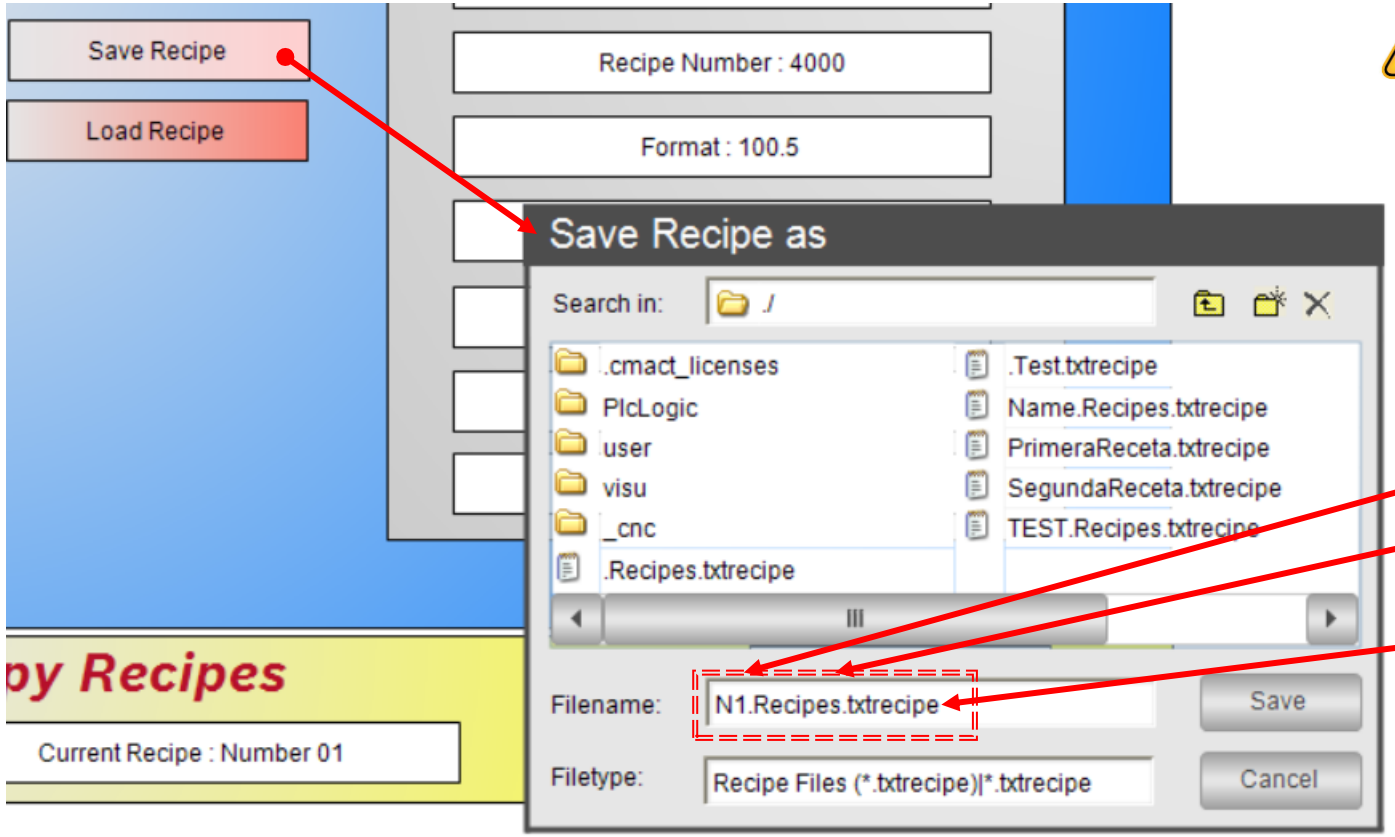
The command in question has two parameters available. The first one, and if you remember what we saw in the previous section, is the "Recipe Definition Name" and the second one is the Recipe that we just created "N1"

Configured Command:

OnMouseClick	Configure...
Execute Command	LoadWriteRecipe, 'Recipes', 'N1'

rexroth
A Bosch Company

- In "Online" and activate the "Save Recipe" button, you will see that a selection screen is displayed. Where we can enter the name of the recipe we want to store



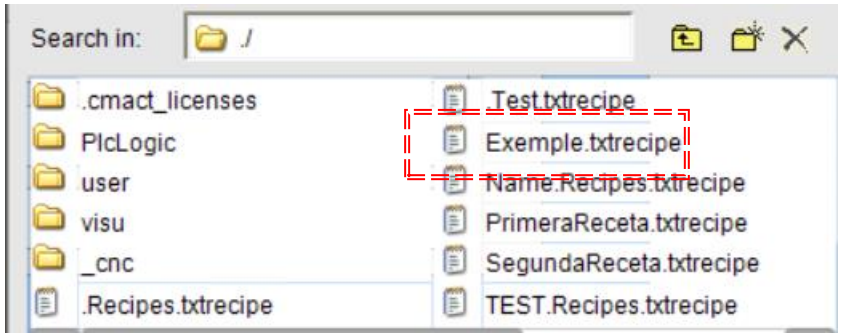
The rest of the buttons should be created in the same way

OnMouseClick	Configure...
Execute Command	SaveRecipeAs, 'Recipes', 'N1'

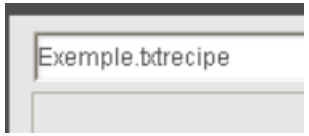
Recipe Name

Definition Name

Extension

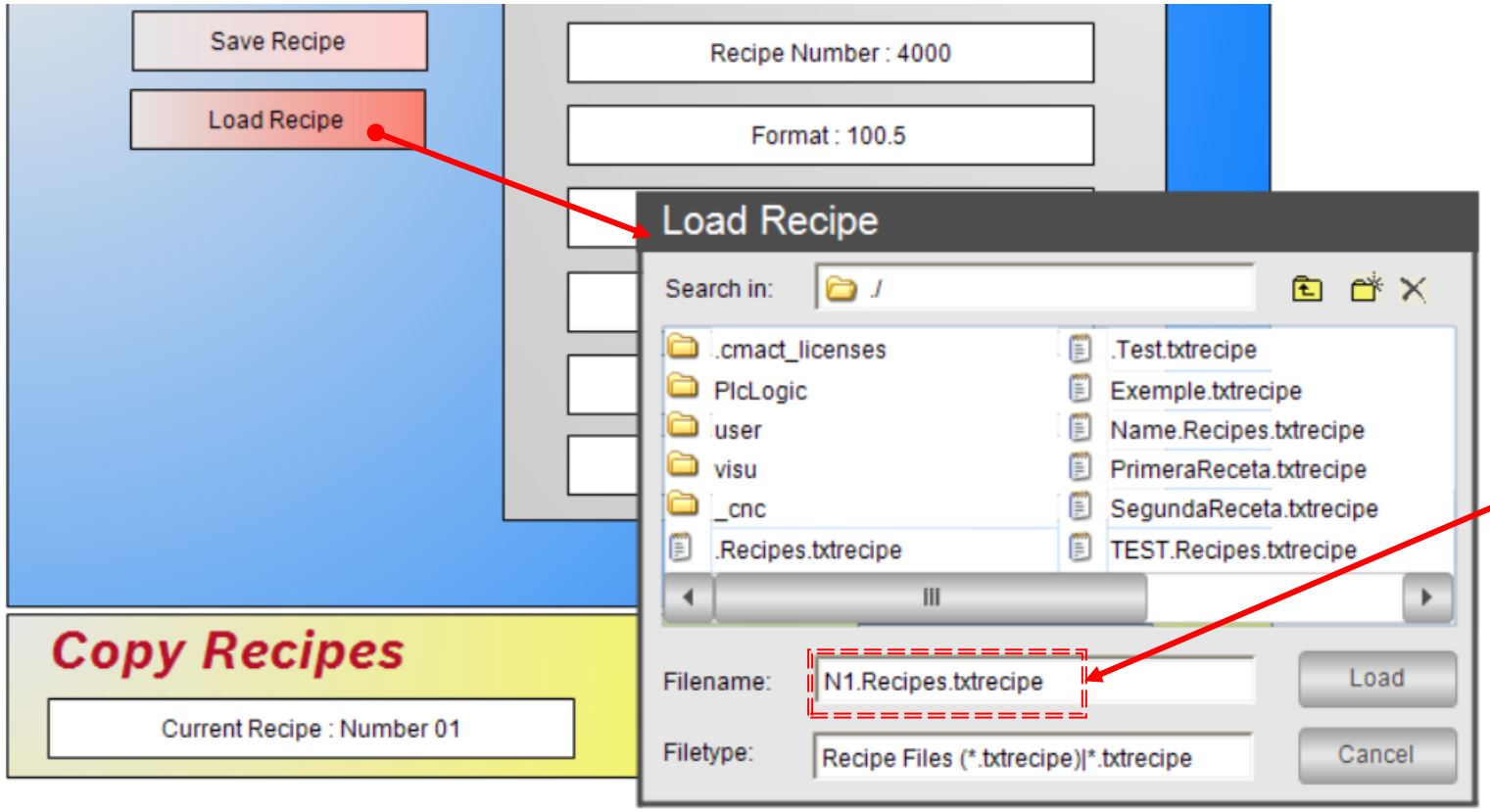


At this point we can use the "Definition Name" or not, since the recipe will be saved anyway



rexroth
A Bosch Company

- With the "Load Recipe" it happens exactly the same and we will be able to load the recipes that we have stored in the PLC variables

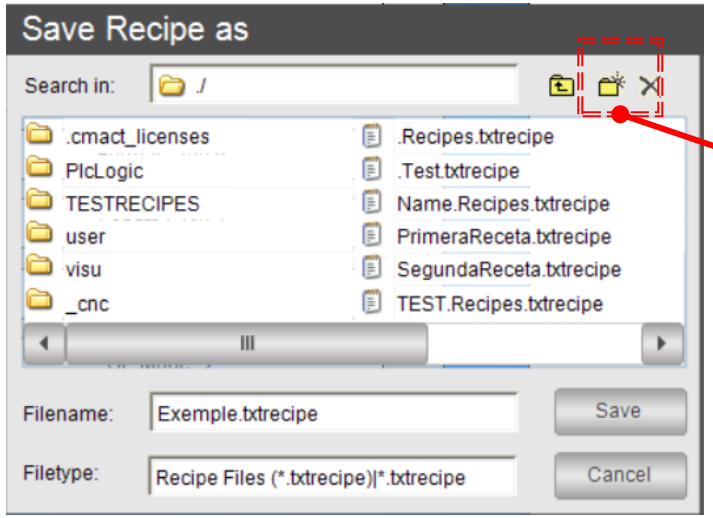


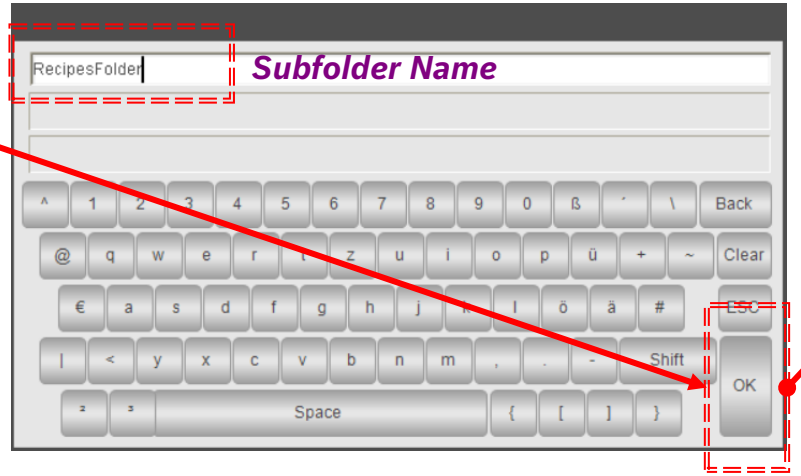
OnMouseClick	Configure...
Execute Command	LoadWriteRecipe, 'Recipes', N1'

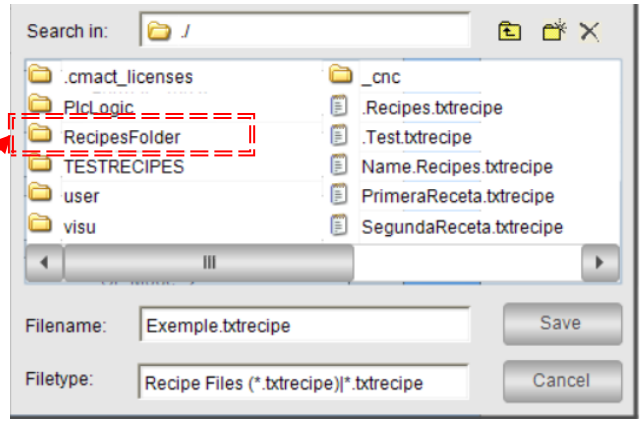
By default, the recipe structure that we had previously defined will be maintained

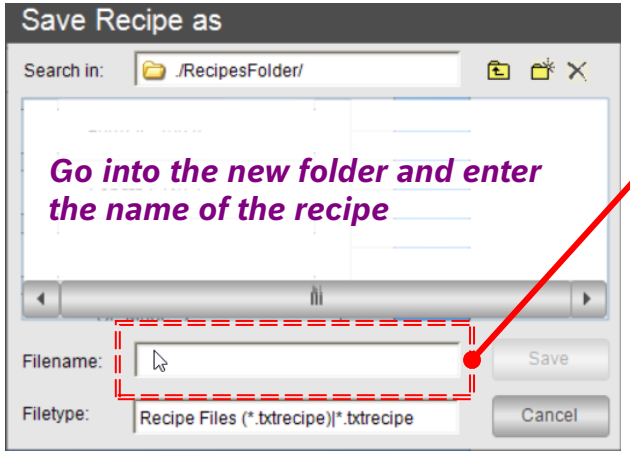
rexroth
A Bosch Company

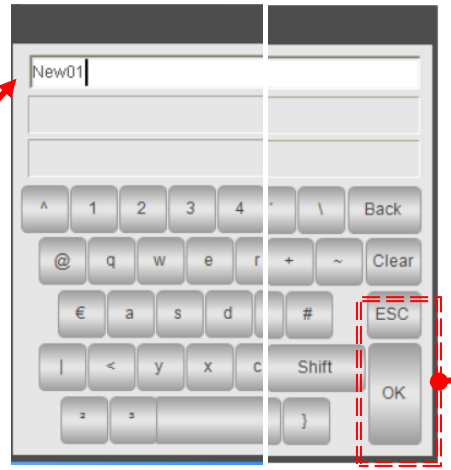
- From my point of view, and in order to facilitate copies of the recipe(s) from or to the outside, such as a USB for example, the best thing would be to generate a specific folder for them.

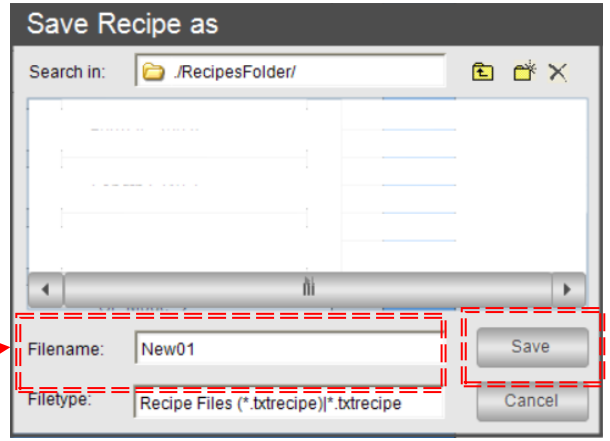


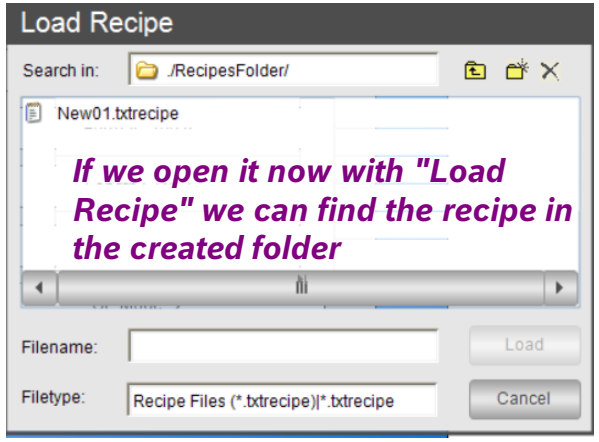












rexroth
A Bosch Company

- List of types of print elements for display :

	Type	Description
Printing integers	%d	Printing a variable (integer data type) as a decimal number
	%i	Printing a variable (integer data type) as a decimal number
	%b	Printing a variable (integer data type) as a binary number
	%o	Printing a variable (integer data type) as an unsigned octal number without a preceding zero
	%x	Printing a variable (integer data type with maximum 32 bits) as an unsigned hexadecimal number without a preceding “0x”
	%u	Printing a variable (integer data type) as an unsigned decimal number
Printing floating-point numbers	%f	In decimal form with decimal point in format 1.6
	%e	Printing a floating-point number (REAL or LREAL) in exponential notation of base 10
Printing text	%c	Printing a single character in ASCII
	%s	Printing a character string
Printing the percent sign	%%	Printing the percent sign in a character string

Possibly not all of them are there, but the list will help us to visualize the most common ones.
In the example, the %s has been used, which, although it is to print a variable of type "string", is perfectly useful to define the values of the variables.

rexroth
A Bosch Company

- List of types of print elements for display :

Printing the date and time	%t[yyyy]	Years with century (4 digits)
	%t[yy]	Years without century (00-99)
	%t[y]	Years without century (0-99)
	%t[MMMM]	Months as a full name
	%t[MMM]	Months as an abbreviated name
	%t[MM]	Months as a number (01 – 12)
	%t[M]	Months as a number (1 – 12)
	%t[ddddd]	Days of week as a number (1=Monday to 7=Sunday)
	%t[dddd]	Days of week as a full name
	%t[ddd]	Days of week as an abbreviated name
	%t[dd]	Days in month as a number (01 – 31)
	%t[d]	Days in month as a number (1 – 31)
	%t[jjj]	Days in year as a number (001-366)
	%t[HH]	Hours in 24-hour format (00-23)
	%t[hh]	Hours in 12-hour format (01-12)
	%t[mm]	Minutes with leading zeros (00-59)
	%t[m]	Minutes without leading zeros (0-59)
	%t[ss]	Seconds with leading zeros (00-59)
	%t[s]	Seconds without leading zeros (0-59)
	%t[ms]	Milliseconds without leading zeros (0-999)
	%t[us]	For LTIME variables only: microsecond definition (0-999)
	%t[ns]	For LTIME variables only: nanosecond definition (0-999)
	%t[t]	If the value is a time < 12h, then A is printed; otherwise P is printed.
	%t[tt]	If the value is a time < 12h, then AM is printed; otherwise PM is printed.
	%t[' ']	If character strings should be printed that correspond to a format definition, then these must be represented in single straight quotation marks.
	%t[yyyy-MM-dd dddd]	Printing the date and day of the week

Recipe Control from the PLC

- The first step is to ensure that we have the two structures created above

```
2  VAR_GLOBAL PERSISTENT RETAIN
3  // Exemple for Recipes Test
4  // Recipe Definition
5     dutRecipePLC: DUT_Recipes;
6  END_VAR
```

Prog10_SeqCtrlRecipe (PRG) → `dutRecipeValues : DUT_Recipes;`

- Next, we will proceed to insert the "Recipe Management" library

RecipeManagement = Recipe Management, 4.1.0.0 (System) Recipe_Management 4.1.0.0

- *Recipe Management, 4.1.0.0 (System)*
  - General Types
  - Interfaces
  - Utilities
  - ReturnValues
  - RecipeManCommands ← ***Modulo general para el control de comandos***
  - TL_RecipeManager

rexroth A Bosch Company

- List of Items in the Recipe Control Module

- General Types
  - InfoValues (ENUM)
  - MessageBox Struct (STRUCT)
- Interfaces
  - IPersistantRecipeListSupportsAdd (ITF)
  - IRecipeCheckOnStart (ITF)
  - IRecipeDefinition2 (ITF)

- RecipeManCommands (FB)
  - RecipeManCommands.CreateRecipe (METH)
  - RecipeManCommands.CreateRecipeNoSave (METH)
  - RecipeManCommands.DeleteRecipe (METH)
  - RecipeManCommands.DeleteRecipeFile (METH)
  - RecipeManCommands.GetLastError (METH)
  - RecipeManCommands.GetLastInfo (METH)
  - RecipeManCommands.GetRecipeCount (METH)
  - RecipeManCommands.GetRecipeNames (METH)
  - RecipeManCommands.GetRecipeValues (METH)
  - RecipeManCommands.GetRecipeVariableNames (METH)
  - RecipeManCommands.LoadAndWriteRecipe (METH)
  - RecipeManCommands.LoadFromAndWriteRecipe (METH)
  - RecipeManCommands.LoadRecipe (METH)
  - Private
  - RecipeManCommands.ReadAndSaveAs (METH)
  - RecipeManCommands.ReadAndSaveRecipe (METH)
  - RecipeManCommands.ReadAndSaveRecipeAs (METH)
  - RecipeManCommands.ReadRecipe (METH)
  - RecipeManCommands.RegisterDatasourceRecipeDefinition (METH)
  - RecipeManCommands.ReloadRecipes (METH)
  - RecipeManCommands.ResetLastError (METH)
  - RecipeManCommands.ResetLastInfo (METH)
  - RecipeManCommands.SaveRecipe (METH)
  - RecipeManCommands.SetRecipeValues (METH)
  - RecipeManCommands.SetStoragePath (METH)
  - RecipeManCommands.WriteRecipe (METH)

Used in the example

In the example only two of the available methods have been used, but that will suffice to handle the issue of recipes.

- ReturnValues (GVL)
- TL_RecipeManager (Text List)
- Utilities
  - RecipeMan_FctTypeClassToDataType (FUN)

Indices and tables

- File and Project Information
- Library Reference

rexroth
A Bosch Company

- Possible errors in method status data:

Name	Type	Initial	Comment
ERR_OK	UDINT	16#0	The operation succeeded
ERR_FAILED	UDINT	16#1	The operation failed
ERR_PARAMETER	UDINT	16#2	Wrong parameter
ERR_NOTINITIALIZED	UDINT	16#3	The dataserver object is not initialized. The dataserver is necessary if the recipe management is used in combination with the CODESYS HMI.
ERR_NOTIMPLEMENTED	UDINT	16#C	The dataserver does not implement the interface IDataServer4, which is necessary if the recipe management is used in combination with the CODESYS HMI.
ERR_NO_OBJECT	UDINT	16#10	Not all recipe definition variables can be written via the dataserver. Only the valid recipe definition variables are written.
ERR_NOMEMORY	UDINT	16#11	The dataserver did not get enough memory
ERR_RECIPE_FILE_NOT_FOUND	UDINT	16#4000	The recipe file was not found.
ERR_RECIPE_MISMATCH	UDINT	16#4001	The content of the recipe file does not match the current recipe. This error is only generated when the storage type is textual (see "RecipeManager" "Storage Type" in your project) and when a variable name in the file does not match the variable name in the recipe definition. The recipe file is not loaded when this error occurs. Possible reasons: ▪ A variable has been removed in the recipe definition of the project.
ERR_RECIPE_SAVE_ERR	UDINT	16#4002	The save operation failed. Possible reasons: ▪ The file cannot be created or opened because the disk is full. ▪ The configured file path does not exist (see "RecipeManager" "File Path" in your project). ▪ The configured file extension is not allowed by the runtime (see "RecipeManager" "File Extension" in your project).
ERR_RECIPE_NOT_FOUND	UDINT	16#4003	The recipe does not exist
ERR_RECIPE_DEFINITION_NOT_FOUND	UDINT	16#4004	The recipe definition does not exist
ERR_RECIPE_ALREADY_EXIST	UDINT	16#4005	The recipe already exists in the recipe definition. Use another name to create a new recipe.

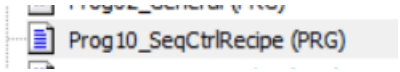
rexroth
A Bosch Company

- Possible errors in method status data:

ERR_NO_RECIPE_MANAGER_SET	UDINT	16#4006	The global recipe manager is not set. Possible reasons: This can happen when the option recipe managem ent in plc is not set in the recipe manager of the current application.
ERR_RECIPE_NOT_ALL_VARIABLES_WERE_LOADED	UDINT	16#4007	The recipe definition contains more variables than the recipe file. In this case the variable values from the recipe file are written anyway. This is only an info not an error.
ERR_RECIPE_NOMEMORY	UDINT	16#4008	The recipe definition has no free memory to create an new recipe. Possible reasons: This can happen when the option "save recipe changes to recipe files automatically" is not set in the recipe manager of the current application. In this case only 50 recipes per recipe definition are possible. If the option "save recipe changes to recipe files automatically" is set the error cannot happen. When the disk is full the error ERR_RECIPE_SAVE_ERR is created.
ERR_RECIPE_MANAGER_LOCKED_DURING_ONLINE_CHANGE	UDINT	16#4009	The recipe manager was locked during online change. Possible reasons: Some of the recipe man commands should be executed while an online change occurs. These commands were not executed during online change.
ERR_SOURCE_EXHAUSTED	UDINT	16#40A0	Used for UTF8 helper
ERR_TARGET_EXHAUSTED	UDINT	16#40A1	Used for UTF8 helper
ERR_SOURCE_ILLEGAL	UDINT	16#40A2	Used for UTF8 helper

rexroth
A Bosch Company

- In the example program, we used the following module for recipe management:



Variables Used for System Control

The truth is that finding the right solution for the operation of what I want to do has resulted in a nightmare of enormous proportions. It is possible that it is my fault, because I do not understand the general operation of the modules or simply because the explanations of the methods and their uses are quite incomprehensible.

Sequence

FB Recipe Management

FB Recipe Delete

Recipe Structure

Dword with possible errors in recipe methods

Control bits to activate the functions of the Recipe

Strings with the definition name, the assigned path of the recipes, and the extension of the files used

String used to define a new recipe

Timers and R_trig assigned to steps

String for recipe delete

```
PROGRAM Prog10_SeqCtrlRecipe
VAR
  // Sequence for Recipe Control
  iSeqCtrlRecipe        :   INT;

  // Recipe Management Fb (For Create, Save and Load Recipe)
  fbRecipeControl       :   RecipeManCommands;

  // Recipe Delete (Using IL_FileDeleteAsync)
  fbRecipeDelete: IL_FileDeleteAsync;

  // Recipe Structure
  dutRecipeValues       :   DUT_Recipes;

  // Output Modul con errors control
  dwRecipesGen          :   DWORD;

  // Recipes Control Bits
  bStartCreateRecipe            : BOOL;
  bStartSaveRecipeValues        : BOOL;
  bStartLoadRecipeValues        : BOOL;
  bStartDeleteRecipe            : BOOL;
  bRecipeToPLC                  : BOOL;
  bPLCToRecipe                  : BOOL;

  // Definition Name
  strRecipeDefName          :STRING := 'Recipes'; // Definition Name
  strPathRecipes            :STRING := 'TestRecipes/'; // Recipe Path
  strFileExtension          :STRING := '.txtrecipe';  // Recipe Extension

  // New Recipe for Create
  strNewRecipe              :STRING;

  // Array for Ton Control Step And R_Trig control Step
  tonCase:ARRAY [0..50] OF TON;
  rtCase:ARRAY [0..50] OF R_TRIG;

  // String to Delete Recipe
  strFileDelete: STRING(255);
END_VAR
```


rexroth
A Bosch Company

- If you look at the methods you will determine that each one has its function and that they should work without further problems, however, "they are difficult to understand", since they seem to act differently from what you want to do, thinking that the use should be able to be managed from the PLC.

1º In most cases, the recipes should be "inserted" in the generated recipe module as in the case of recipe N1, so it would be necessary to "insert" them by hand beforehand, which is a problem, if we want to execute it from the screen and without the intervention of the operator in this part of the design.

I'm not a big fan of "folding" when I have a problem or something the system seems determined not to let me do. That's why I always say that there is some hidden solution and we must find it, even if it's by searching and trying and testing

Variable	Type	Name	Comm...	Minim...	Maxim...	Curren...	N1	N2	N3
Prog10_SeqCtrlRecipe									
dutRecipeValues									
strRecipeName	STRING								
strRecipeNum	STRING								
rFormat	REAL								
rlength	REAL								
rWidth	REAL								
iOpeMode	INT								
bSelFunc	BOOL								

2º If the Recipes are not fixed in the "Recipe", with the "create" for example it generates the *.txtrecipe files but they are subject to certain conditions and once the power stops and it starts up again it seems that internally the link between Recipe and recipes disappears by magic and what worked before, Now it's gone.

3º The GetRecipeNames option, which returns all the recipes created and which is very interesting, had to be discarded, since when removing tension and restarting the array was empty and the information was lost despite the fact that the files were still present in the folder.

4º As I have already said, it has been chaotic to find a moderately "solid" solution to be able to manage the system as it was presented from the beginning. As you can see, the final recipe control program is short. Although it is necessary to add a part for the control of existing files, the function that should be used with "GetRecipeNames" and that will be executed in another way

- The sample program is as follows: (It's just a simple example, which should surely be improved)

If we take action Load Values To PLC

The values of the selected recipe will be loaded into the Job Values (PLC)

```
// Send Values From Recipe Selected To PLC Values
IF bRecipeToPLC THEN
    dutRecipePLC:= dutRecipeValues;
END_IF
```

Technically, this option should be "protected" and only run if we are, for example, in reformatting mode.

Start of the Case on which we will move using the associated buttons

Create Recipe

Delete Recipe

Save Recipe

Load Recipe

Load PLC Values To Recipe

```
// Control Sequence of Recipe Control
CASE IseqCtrlRecipe OF
0: // Waiting to Start
IF bStartCreateRecipe THEN // Sequence Start To Create Recipe
        IseqCtrlRecipe:= 1;
ELSIF bStartDeleteRecipe THEN // Sequence Start To Delete Recipe
        IseqCtrlRecipe:= 2;
ELSIF bStartSaveRecipeValues OR bPLCToRecipe  THEN // Sequence Start To Save Recipe
        IseqCtrlRecipe:= 3;
ELSIF bStartLoadRecipeValues THEN // Sequence Start To Load Recipe
        IseqCtrlRecipe:= 4;
END_IF
```


rexroth
A Bosch Company

In case 1, we created the recipe. But be careful, we are not using the CreateRecipe, but the method that creates a file based on the "Definition Name" and the N1 recipe that we had already created previously.

Timers are used to manage the steps.

Create Recipe

```
1: // Create Recipe
dwRecipesGen:=fbRecipeControl.ReadAndSaveRecipeAs(strRecipeDefName,'N1',strNewRecipe);

// Control Time to change Step
IF tonCase[1].Q THEN
IseqCtrlRecipe:= 0;
END_IF
```

```
strRecipeDefName        :STRING := 'Recipes'; // Definition Name
```

Formato del archivo creado

file888.txtrecipe

Case 2 is a strange case. For starters, it doesn't have a code and it does have the control timer. It is used to "Delete" the recipe and since as in other cases the DeleteRecipe does not seem to work with the file, the solution has been to use the file deletion that does work without problems.

Delete Recipe

```
2: // Delete Recipe


// Control Time to change Step
IF tonCase[2].Q THEN
IseqCtrlRecipe:= 0;
END_IF
```

Activating step 2 (Delete) activates the file deletion module from the file obtained from the file list (which we will see later) and obtaining the access structure of the two lines that use the CONCAT command

```
// CONCAT to Path extract of delete file
strFileDelete:= CONCAT(strPathRecipes,arFilesVisu[iPosArray]);
strFileDelete:= CONCAT (strFileDelete,strFileExtension);
// FB Recipe delete control
fbRecipeDelete( //IL FileDeleteAsync
    Execute:=(iSeqCtrlRecipe=2) ,
    Done=> ,
    Active=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    FileName:=ADR(strFileDelete) );
```

Paso 2

The position pointer will be sent to us by the selection of the display object.

Ruta del archivo a Borrar

Para poder usarlo deberéis de instalar la librería CXA_FILEASYNC

CXA_FILEASYNC = CXA_FileAsync, 1.20.0.2 (Bosch Rexroth AG)

rexroth A Bosch Company

In case 3, we save the recipe, if you look you will see that the same module as case 1, only in this case we do not choose a file by hand but from the list that I have already mentioned that we will see later.


```
3: // Save Recipe

IF bPLCToRecipe THEN
dutRecipeValues:= dutRecipePLC;
END_IF
dwRecipesGen:=fbRecipeControl.ReadAndSaveRecipeAs(strRecipeDefName,'N1',arFilesVisu[iPosArray]);

// Control Time to change Step
IF tonCase[3].Q THEN
IseqCtrlRecipe:= 0;
END_IF
```

This manoeuvre allows the PLC data to be written to the Recipe, but only if that button is activated.

```
strRecipeDefName        :STRING := 'Recipes'; // Definition Name
```

The position pointer will be sent to us by the selection of the display object.

In case 4, we load the recipe and, as in the previous case, we choose the file from the list of files that we will have read from the files folder


```
4: // Load Recipe From Files
dwRecipesGen:=fbRecipeControl.LoadFromAndWriteRecipe(strRecipeDefName,'N1',arFilesVisu[iPosArray]);

// Control Time to change Step
IF tonCase[4].Q THEN
IseqCtrlRecipe:= 0;
END_IF

END_CASE
```

The position pointer will be sent to us by the selection of the display object.

rexroth
A Bosch Company

Finally, outside the Case, we have timers one per step and that are executed with the step activated.

```
// Timers Control change Step
tonCase[0](IN:=(IseqCtrlRecipe=0),PT:=T#100MS);
tonCase[1](IN:=(IseqCtrlRecipe=1),PT:=T#100MS);
tonCase[2](IN:=(IseqCtrlRecipe=2),PT:=T#100MS);
tonCase[3](IN:=(IseqCtrlRecipe=3),PT:=T#100MS);
tonCase[4](IN:=(IseqCtrlRecipe=4),PT:=T#100MS);
tonCase[5](IN:=(IseqCtrlRecipe=5),PT:=T#100MS);
tonCase[6](IN:=(IseqCtrlRecipe=6),PT:=T#100MS);
tonCase[7](IN:=(IseqCtrlRecipe=7),PT:=T#100MS);
tonCase[8](IN:=(IseqCtrlRecipe=8),PT:=T#100MS);
tonCase[9](IN:=(IseqCtrlRecipe=9),PT:=T#100MS);
```

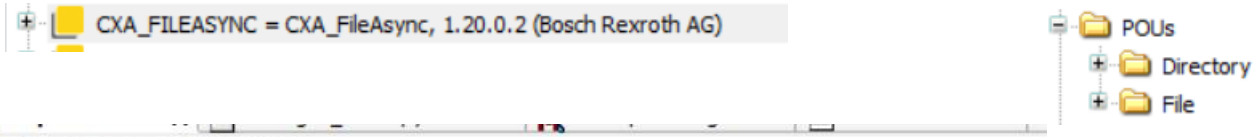
- In short, the control program itself is not complicated at all, only that on many occasions the methods did not meet expectations and the solution had to be sought elsewhere

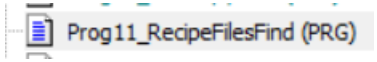
But as I say, you always have to keep investigating, because the solution, or at least one of them, is hidden somewhere.

rexroth
A Bosch Company

- The next topic is, how do we find out what files we've been creating and visualize them consistently?

The previously installed library contains, in addition to file control, also directory control





Variables utilizadas para el control del sistema

Bits and integer of the file read control sequence

FB control to determine the existing files in the directory

File Reread Control Bit

Array that resets the search array to "clean" the structure if it has been modified

Timers and R_trig assigned to steps

Variables for Controlling and Eliminating Gaps in the Recipes Viewer

```
PROGRAM Prog11_RecipeFilesFind
VAR
    // Sequence Control
    bStartSeq        : BOOL; // Initialization Sequence
    bInitSeq         : BOOL; // Init activated only in the firts start
    iSeqReadFiles    : INT;  // Sequence control

    fbDirRead: IL_DirReadAsync; ////Control Modulo for read File in Directory

    bReadNext: BOOL; // Bit for Start New Read File

    arFilesFindAux: ARRAY [0..10] OF STRING; // Structure To Reset array of files find

    // Array for Ton Control Step And R_Trig control Step
    tonCase: ARRAY [0..10] OF TON;
    rtCase: ARRAY [0..10] OF R_TRIG;

    // Variables to control the elimination of gaps in the array of recipes shown
    iPos: INT;
    ILen: INT;
    arFilesVisuAux: ARRAY [0..10] OF STRING;
    x: INT;
    y: INT;

END_VAR
```


rexroth
A Bosch Company

- ***In the sample program, the reading of the files is managed with a total of four steps.***

In Case 0, the reset of the new read bit is generated and one of the signals for boot is expected to be activated:

```
// Sequence to Files Read
CASE iSeqReadFiles OF
0: // Init Sequence
bReadNext:= FALSE;

// Start Init Sequence after start de PLC Program
IF NOT bInitSeq THEN
    bStartSeq:=TRUE;
ELSE
    bStartSeq:=FALSE;
END_IF

// Sequence Start and Inicialization of the Array control Read Files
IF (bStartSeq OR Prog10_SeqCtrlRecipe.bStartCreateRecipe OR Prog10_SeqCtrlRecipe.bStartDeleteRecipe)  THEN
    arFilesFind:=arFilesFindAux;
    iSeqReadFiles:=1;
END_IF
```

Reset of the read the file name bit

Sequence start-up control after a new PLC start-up

Run part for updating the viewer during the first boot and if the Create Recipe or Recipe Delete buttons are activated

- In Case 1 we wait until the system does not indicate that this "InOperation" in this step "cleans" the previous values of the control module and initializes the receipt of the file names:

```
1: // Check Modul Read Directiry in Operation

IF fbDirRead.InOperation AND tonCase[1].Q THEN
    iSeqReadFiles:=2;
END_IF
```

The control module is located outside the case and at the end of the POU

rexroth
A Bosch Company

- ***In Case 2, the reread bit is reset again and the value received from the file module is copied into the file array that we are going to use next***

```
2: // Read Files from Folder
bReadNext:= FALSE;
arFilesFind[fbDirRead.EntryCnt]:=fbDirRead.DirEntry.EntryName;

IF tonCase[2].Q THEN
    iSeqReadFiles:=3;
END_IF
```

File name (name.extension)

Used as a pointer, we can modify the number of arrays we want to write about.

The control module is located outside the case and at the end of the POU.
In this case:
fbDirRead.EntryCnt tells us the current number of files found in the directory.
fbDirRead.DirEntry.EntryName shows us the name of the file, including the extension or any other element that composes it

This is the structure we are receiving

arFilesFind	ARRAY [0..10] OF ...	
arFilesFind[0]	STRING	''
arFilesFind[1]	STRING	'.txtrecipe'
arFilesFind[2]	STRING	'Part004.txtrecipe'
arFilesFind[3]	STRING	'Part007.txtrecipe'
arFilesFind[4]	STRING	'Part003.txtrecipe'
arFilesFind[5]	STRING	'Part002.txtrecipe'
arFilesFind[6]	STRING	'file888.txtrecipe'
arFilesFind[7]	STRING	'..'
arFilesFind[8]	STRING	'.'
arFilesFind[9]	STRING	'Part001.txtrecipe'
arFilesFind[10]	STRING	''

This variable is located in a general variables folder

```
arFilesFind: ARRAY [0..10] OF STRING;
```


rexroth
A Bosch Company

- ***In Case 3, the reread command is triggered and the reread acknowledgment is awaited before returning to Case 2 and copying a new received file name.***

```
3: // Next File Read and control error of the module
bReadNext:= TRUE;

IF fbDirRead.ReadNextDone THEN
    iSeqReadFiles:=2;
END_IF

IF fbDirRead.Error THEN
    iSeqReadFiles:=0;
    bInitSeq:=TRUE;
END_IF

END_CASE
```

The sequence returns to Case 2 as long as there is no error and we read the files located in the folder

We use the module's error bit to exit the sequence. Normally, when the system reaches the last file, it generates a crash and we take the opportunity to end the reading process.

- ***Outside of the Case, we have the timers used to jump to the next step***

```
// Timers Control change Step
tonCase[0](IN:=(iSeqReadFiles=0),PT:=T#500MS);
tonCase[1](IN:=(iSeqReadFiles=1),PT:=T#500MS);
tonCase[2](IN:=(iSeqReadFiles=2),PT:=T#500MS);
tonCase[3](IN:=(iSeqReadFiles=3),PT:=T#500MS);
tonCase[4](IN:=(iSeqReadFiles=4),PT:=T#500MS);
```


rexroth
A Bosch Company

- *And also the file reading control module*

```
// Read Files of Folder 'TestRecipes'
fbDirRead( //IL_DirReadAsync
    Enable:=(iSeqReadFiles > 0) ,
    DirName:=ADR('TestRecipes/') ,
    ReadNext:= bReadNext,
    InOperation=> ,
    Active=> ,
    ReadNextDone=> ,
    Shutdown=> ,
    Error=> ,
    ErrorID=> ,
    ErrorIdent=> ,
    EntryCnt=> ,
    DirEntry=> );
```

- *If the sequence is activated, the module enters "inOperation"*
- *Path where the files are located*
- *Bit for read new file name control*
- *Module Activated Correctly*
- *Next file Read Bit done*
- *Read File Counter*
- *Structure of the read file*

DirEntry	IL_DIR_INFO	
EntryName	STRING(500)	'Part001.txtrecipe'
EntrySize	UDINT	16#00000177
EntryTime	IL_DIR_FILE_TIME	
Creation	DATE_AND_TIME	DT#2023-2-3-15:24:57
LastAccess	DATE_AND_TIME	DT#2023-2-3-15:25:14
LastModification	DATE_AND_TIME	DT#2023-2-3-15:24:57

- *Nombre del archivo*

rexroth
A Bosch Company

- ***The last part of the module will be the one that will remove the extension and place the names of the files without holes in the display system.***

```
// Extraction of Valid Recipe Name
iPos:=FIND(arFilesFind[x],'.');

// Name of the existing Recipes in the directory
arFilesVisuAux[x]:=MID(arFilesFind[x],iPos-1,1);

// Discriminate gaps in the results received from the directory
IF arFilesVisuAux[x] <> '' THEN
    arFilesVisu[y]:=arFilesVisuAux[x];
    y:=y+1;
END_IF
// Increase control pointer
x:=x+1;
// Initialization of the variables (should be managed based on the real recipes that we have)
IF x >10 THEN
    x:=0;
    y:=0;
END_IF
```

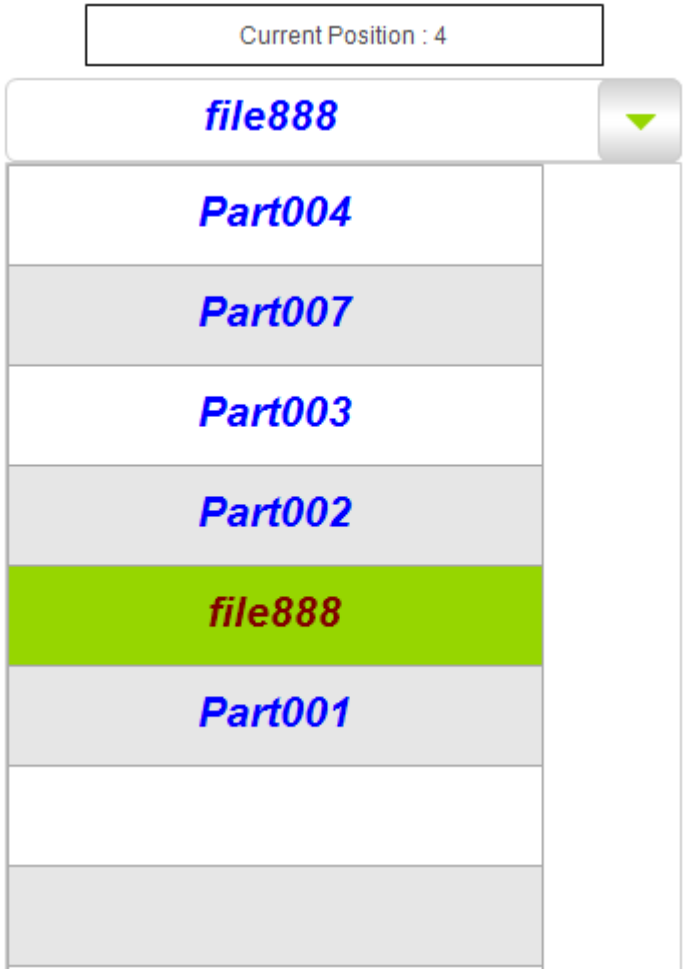
Position in the string of the first dot is located

The received text is cut to show only the name

As the system sends a result with " and to eliminate gaps, the pointer "y" is used, which will only be incremented when a name other than " arrives

Increment of the x-pointer that reads the incoming values

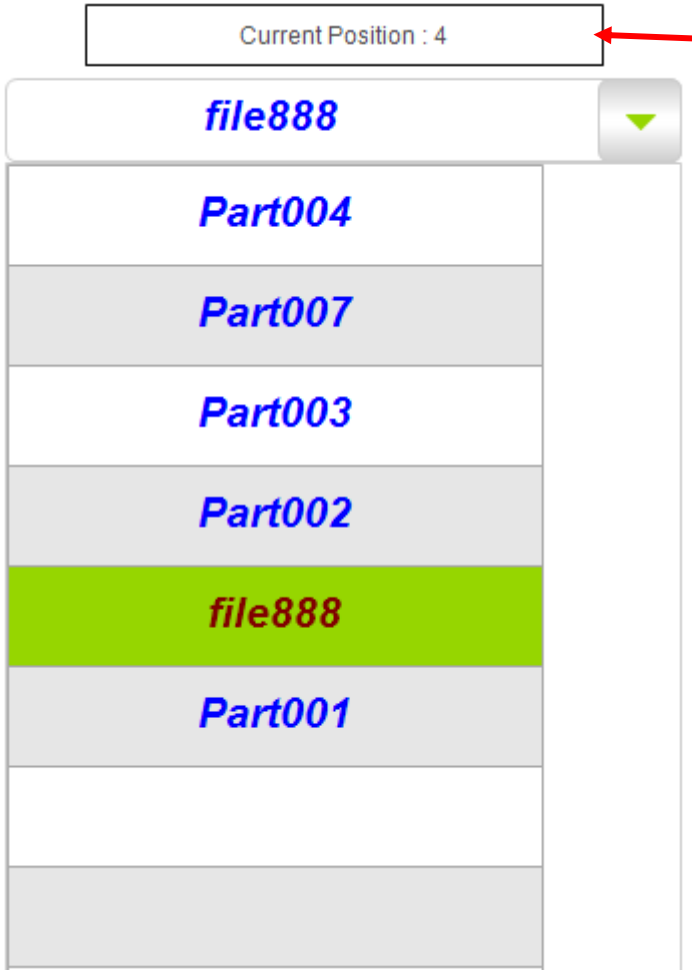
Reset pointers



rexroth
A Bosch Company

- ***The display object must also be programmed.***

Combo Box, Array



Position	
X	32
Y	259
Width	338
Height	42
Variable	iPosArray
Data array	arFilesVisu

Position pointer over the array structure

File Display Array

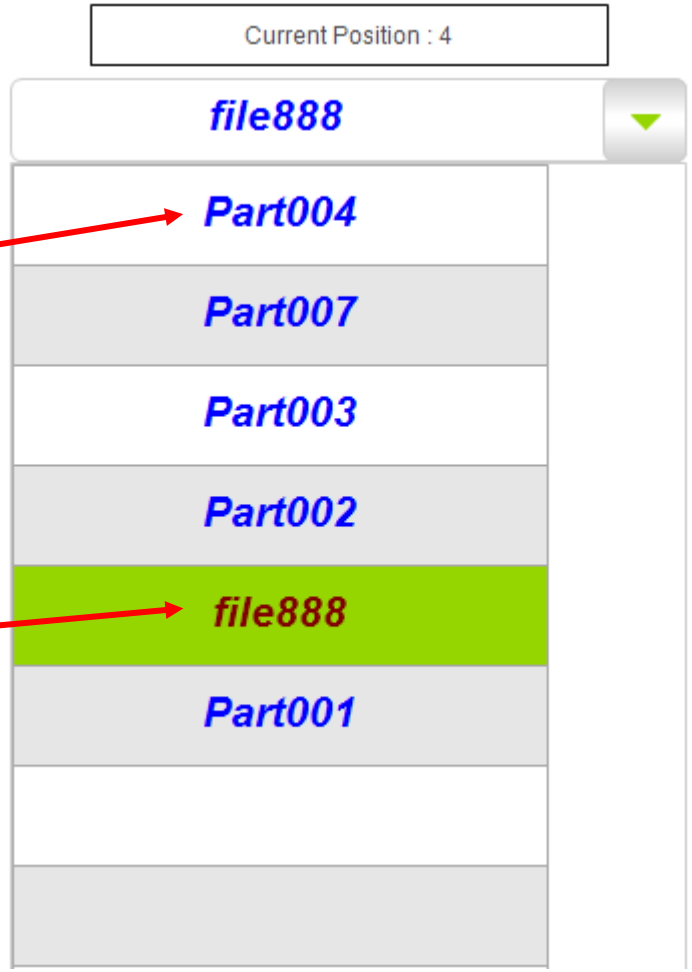
arFilesVisu	ARRAY [0..10] OF ...	
arFilesVisu[0]	STRING	'Part004'
arFilesVisu[1]	STRING	'Part007'
arFilesVisu[2]	STRING	'Part003'
arFilesVisu[3]	STRING	'Part002'
arFilesVisu[4]	STRING	'file888'
arFilesVisu[5]	STRING	'Part001'
arFilesVisu[6]	STRING	''
arFilesVisu[7]	STRING	''
arFilesVisu[8]	STRING	''
arFilesVisu[9]	STRING	''
arFilesVisu[10]	STRING	''

rexroth
A Bosch Company

- *Other parameters used in the display object example:*

Columns	
Column	
[0]	☑
Width	267
Image col...	☐
Image conf...	
Fill mode	Fill cell
Trans...	☐
Trans...	Black
Text align...	Centered
Use templ...	☐
Maximum array index	100
Row height	50
Number of visible rows	100
Scroll Bar size	10

Text properties	
Usage of	Individual settings
Individual text pro...	
Font	*Arial; 16*
Font color	0; 0; 255
Individual font vari...	
Font name	
Size	
Flags	
Character set	
Color	
Individual selection...	
Font	*Arial; 16*
Font color	128; 0; 0
Transpare...	255
Individual selection...	
Font name	
Size	
Flags	
Character set	
Color	

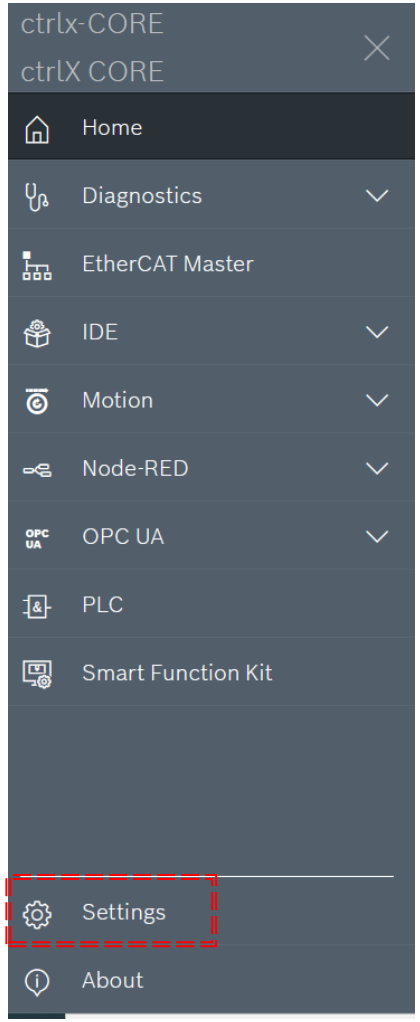


rexroth
A Bosch Company

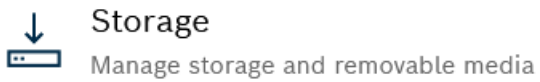
Controlling USBs on a ctrlX Core

- ***In the example we are going to use as a base the USB located in an X3 and specifically the existing one in the XF01 connector***

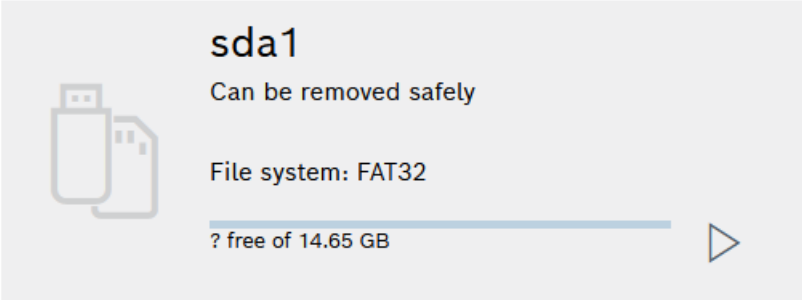
To access the status of the USB, you have to access it from the "Settings" tab of the device to which you are connected.

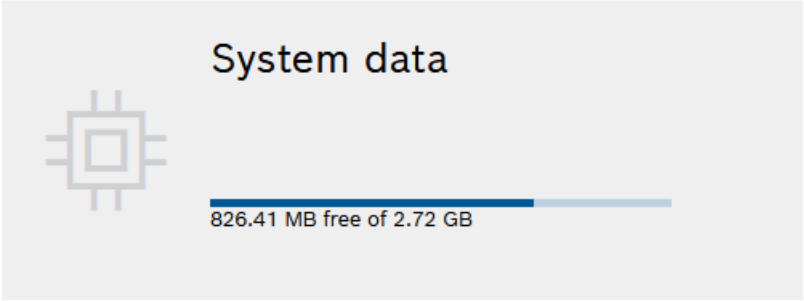


We should display the "Storage" option



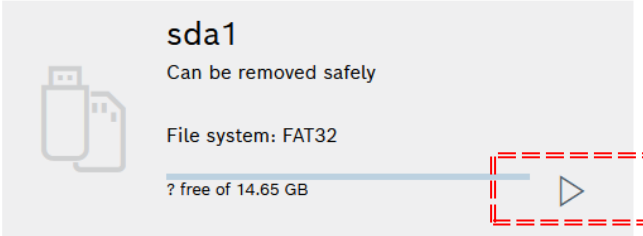
And inside it the USB and System Data:

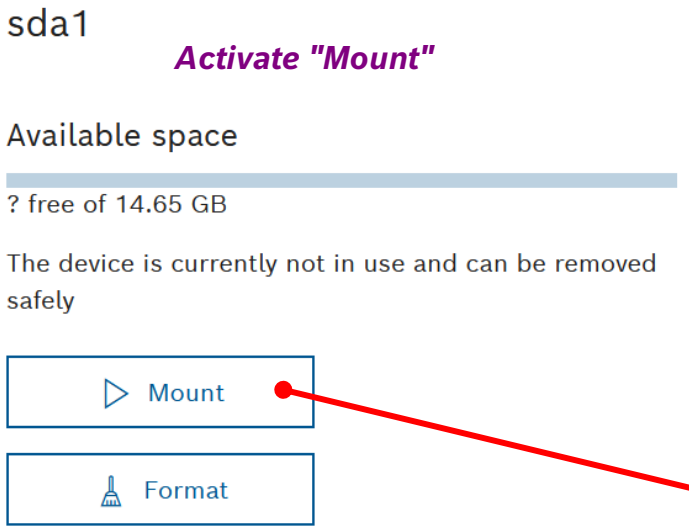


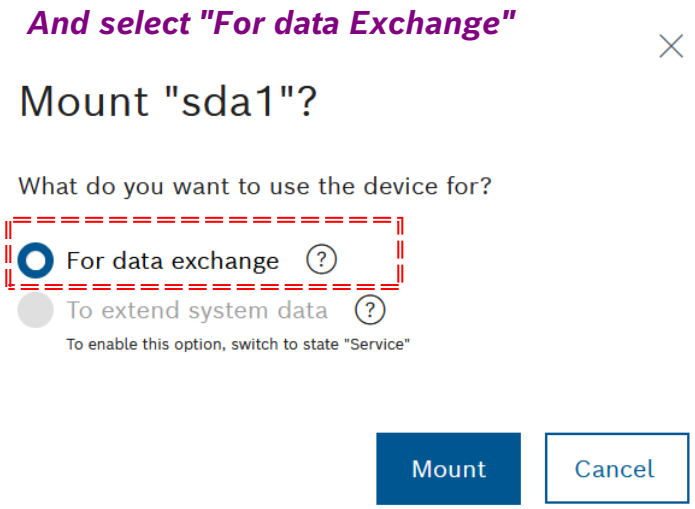


rexroth
A Bosch Company

- ***By default, the USB comes unmounted and should be assembled before it can be used. As you can see, apart from the "Mount" option, there is also the "Format" option***

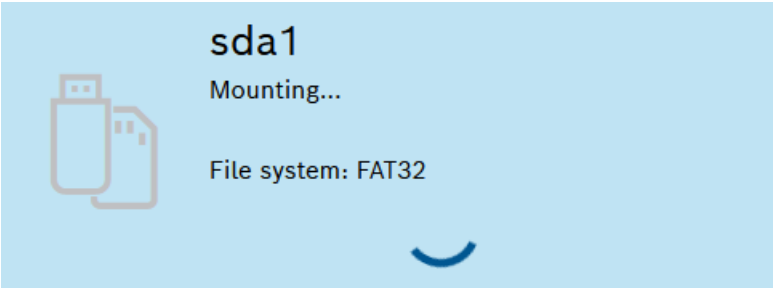


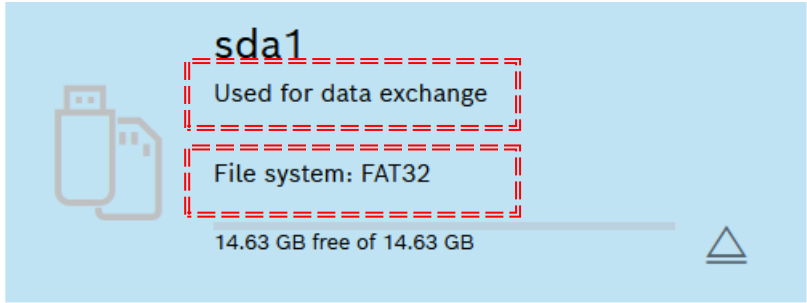


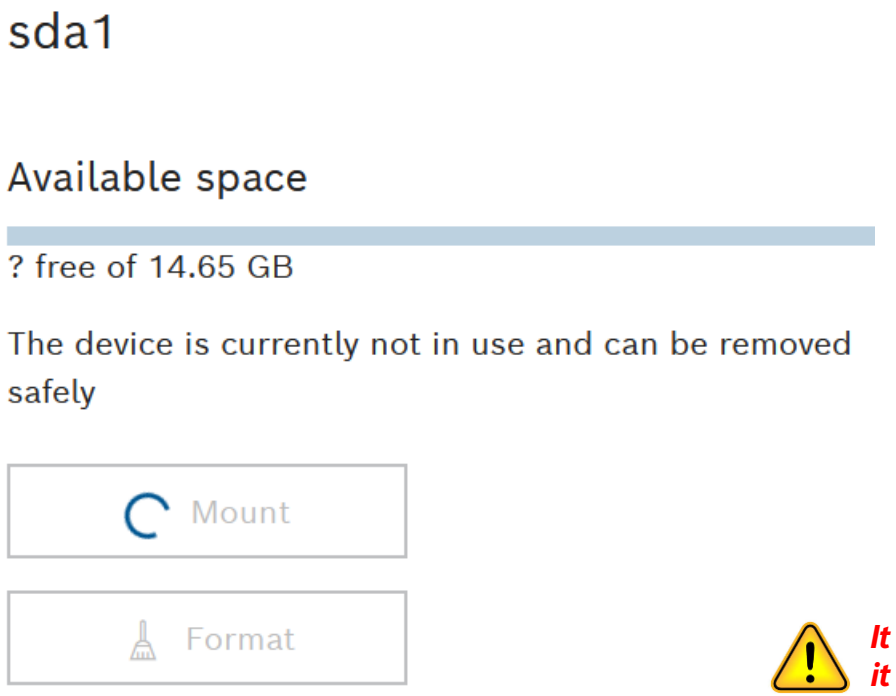


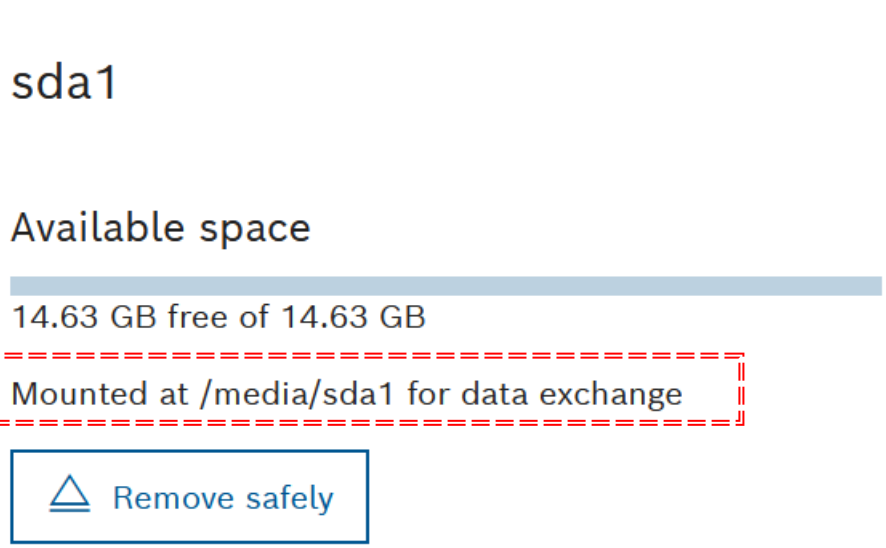
rexroth
A Bosch Company

- ***After a period of waiting, the USB is mounted and ready to use:***









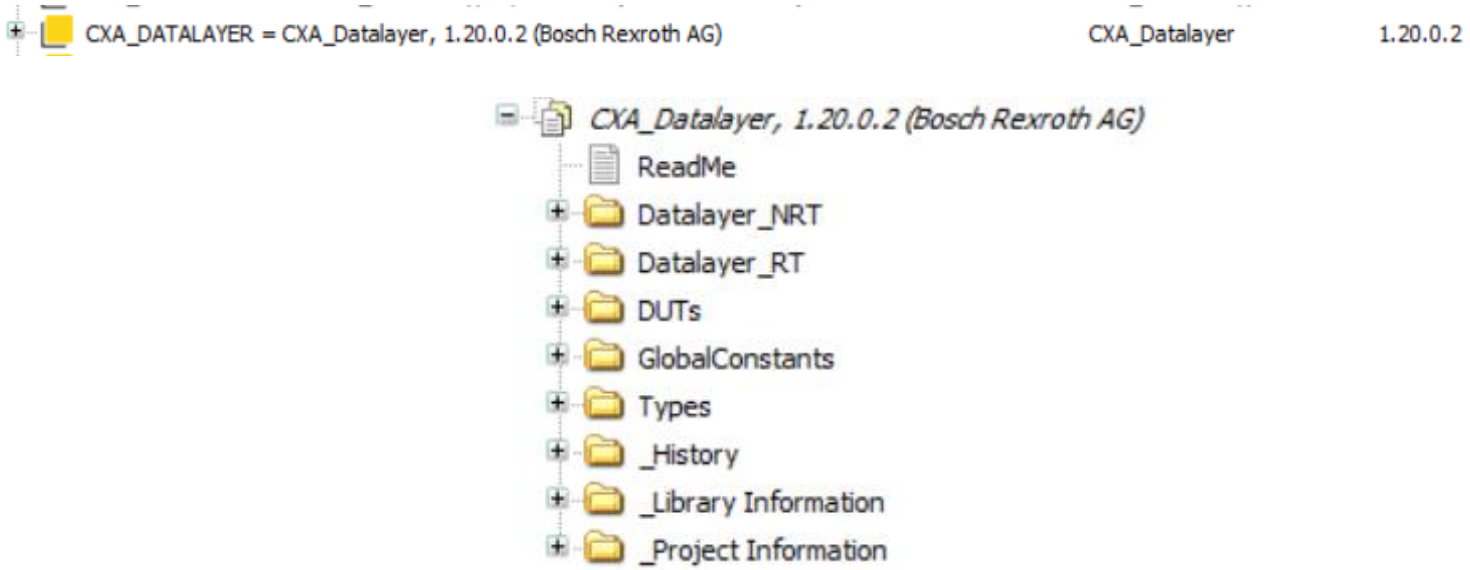
It is recommended to use the USB in FAT32 format, otherwise, it may not be read correctly.

rexroth
A Bosch Company

- ***Obviously, if what we want is to insert the USB and for the system to run itself, that is, to run the "Mount" without having to access the device but by inserting it into the corresponding connector, we must use the following way:***
  - ***On the one hand, PLC functions to determine that the USB is connected***
  - ***On the other hand, NodeRed will have to execute the "Mount" from the commands coming from the PLC***

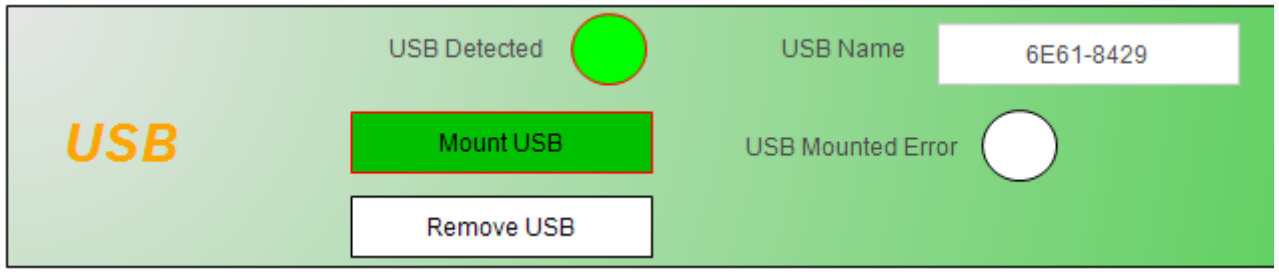
From PLC there is no direct access to the "Mount" button

- ***To do this, we will first insert the DataLayer access library, CXA_DATALAYER, which will allow us to access the different sections of the system.***



rexroth
A Bosch Company

- ***The system will detect the introduction of the USB and activate the rest of the functions to "mount" and leave the USB ready for file transfer.***



- ***The module in charge of controlling the reading of the Datalayer is:***

Prog12_Datalayer (PRG)

Variables Used for System Control

```
VAR
// FB Datalayer control
fbReadDLStorage: CXA_Datalayer.DL_BrowseNode;
fbReadUSBMounted:CXA_Datalayer.DL_ReadNode;

// Used Strings
strNodenamePath      : STRING(255) := 'system/resources/storage/';
strPendriveName      : STRING;
strPathUSBMounted    : STRING;
strNodenamePathNew   : STRING(255) := 'system/resources/storage/storages/uuid';
strUSBName           : STRING;
```

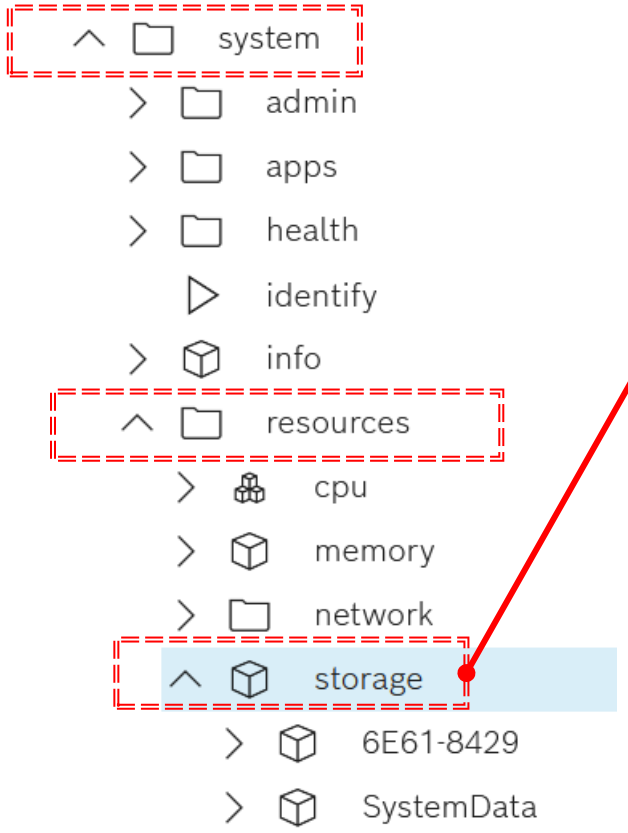
Main function with which we read the node

Function with which we verify that the USB is mounted

```
// Pointer To String
strValue            : POINTER TO POINTER  TO STRING;

// Bits control state USB
bUSBDetect: BOOL;
bMountedOk:BOOL;
bUSBMountedError: BOOL;

// Wait Timer to control state after detect USB
tonUSBMountedOk: TON;

END_VAR
```


rexroth
A Bosch Company

- ***Before we get into the code part, let's see how the structures appear inside the Datalayer with the USB connected in the ctrlX:***



Value (object)

```
1  {
2    "storages": [
3      {
4        "size": 15711830016,
5        "used": 49152,
6        "label": "",
7        "uuid": "6E61-8429",
8        "mounted": true,
9        "format": "fat32",
10       "device": "sda1",
11       "mountPoint": "/media/sda1"
12     },
13     {
14       "size": 2917068800,
15       "used": 2050514944,
16       "label": "ubuntu-data",
17       "uuid": "bb5368fe-5d01-4d42-ba98-fe054da8b295",
18       "mounted": true,
19       "format": "ext4",
20       "device": "mmcblk0p4",
21       "mountPoint": ""
22     }
23   ]
24 }
```

USB Structure

USB Indicator

Estado "Mounted"

System Data Structure

rexroth
A Bosch Company

- *The first part of the code reads the structure discussed before the Datalayer*

```
1   //Read Datalayer From 'system/resources/storage'
2       fbReadDLStorage(
3       Execute:= blTimer100ms.OUT ,
4       Done=> ,
5       Active=> ,
6       Error=> ,
7       ErrorID=> ,
8       ErrorIdent=> ,
9       ClientId:= ,
10      NodeName:=strNodenamePath ,
11      ChildPtr=>strValue ,
12      ChildNum=> );
```

```
strNodenamePath     : STRING(255) := 'system/resources/storage/';
strValue            : POINTER TO POINTER  TO STRING;
```

With the strValue structure, we receive a series of data on a pointer, of which the only ones we should be interested in are the first ones. Once these values have been dereferenced, they show us the name of the USB and the Data System.

```
strValue[0]^ '6E61-8429' ;
strValue[1]^ 'SystemData' ;
```

For practical purposes, the only one that works for what we want to do is the value 0 that corresponds to the USB

rexroth
A Bosch Company

- ***Checking to find out if the USB is connected in this case is easy, since we can only have the USB and the System Data***

```
14  // If the Pendrive is connected, the name is extracted.
15  IF fbReadDLStorage.ChildNum > 1 THEN
16      strUSBName:= strUSBNameDL:= strValue[0]^;
17      bUSBDetect:= TRUE;
18  ELSE
19      strUSBName:= strUSBNameDL:='';
20      bUSBDetect:= FALSE;
21      bMountedOk:= FALSE;
22  END_IF
```

The internal structure of the "ChildNum" changes when the USB is plugged in and goes from being larger than `fbReadDLStorage.ChildNum 2`

At that moment we activate, copy the name of the detected USB to the screen viewer and also the one we will send to the Datalayer and activate the bit "bUSBDetect" that indicates that we have the USB inserted.

Obviously, if we don't have the USB plugged in, the name is deactivated and the detected and mounted USB bits are reset.

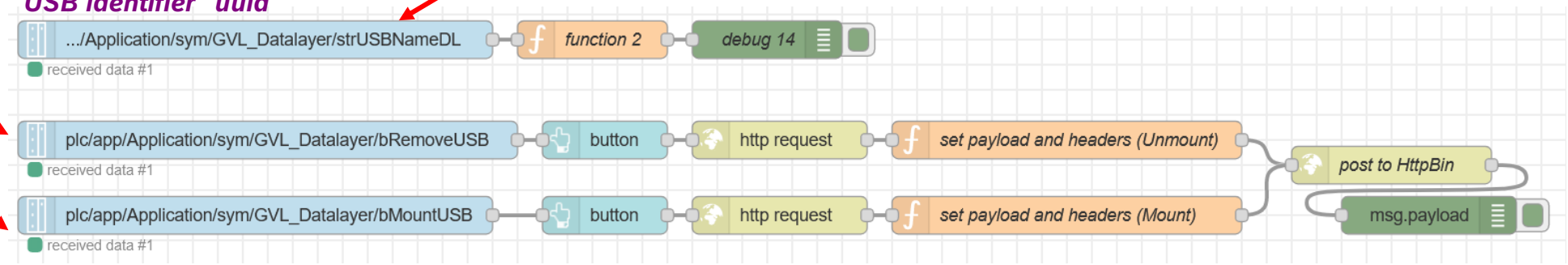
- ***If we detect the USB, we proceed to activate the "Mount" command, which in this case will be executed from NodeRed (we will see the NodeRed part later)***

```
32  // Enable control bits for Mount and remove. These bits activate the NodeRed part
33  bMountUSB:= bUSBDetect;
34  bRemoveUSB:= NOT bUSBDetect;
```

This indicator is important to be able to dynamically activate any USB that we can place.

```
strUSBNameDL:= strValue[0]^;
```

USB identifier "uuid"



rexroth
A Bosch Company

- ***The next step once the Nodered part has been executed is to check that the USB is mounted***

```
8   strNodenamePath     : STRING(255) := 'system/resources/storage/';

36  // Path to see if the USB is "mounted"
37  strPathUSBMounted:= CONCAT (strNodenamePath,strUSBName);
38  strPathUSBMounted:=CONCAT(strPathUSBMounted,'/mounted');
39
40  // Read USB Mount Status
41  fbReadUSBMounted(
42      Execute:=NOT fbReadUSBMounted.Done AND NOT fbReadUSBMounted.Error AND bUSBDetect ,
43      Done=> ,
44      Active=> ,
45      Error=> ,
46      ErrorID=> ,
47      ErrorIdent=> ,
48      ClientId:= ,
49      NodeName:=strPathUSBMounted ,
50      Value:= bMountedOk,
51      Size=> );
52
53  // Wait Time to control state after detec USB
54  tonUSBMountedOk(IN:=bUSBDetect AND NOT bMountedOk,PT:=T#15S);
55  // Mounted USB Error
56  bUSBMountedError:=tonUSBMountedOk.Q  ;
57
```

The USB name will depend on the inserted USB

```
{
    "size": 15711830016,
    "used": 49152,
    "label": "",
    "uuid": "6E61-8429",
    "mounted": true,
    "format": "fat32",
    "device": "sda1",
    "mountPoint": "/media/sda1"
},
```

Full path to the "mounted" state that will depend on the indicator on each USB

Properly mounted USB

Error control during the assembly process

The "Mount" error may vary depending on the characteristics of each USB and it may be the case that an error is indicated prior to indicating the "mount" status

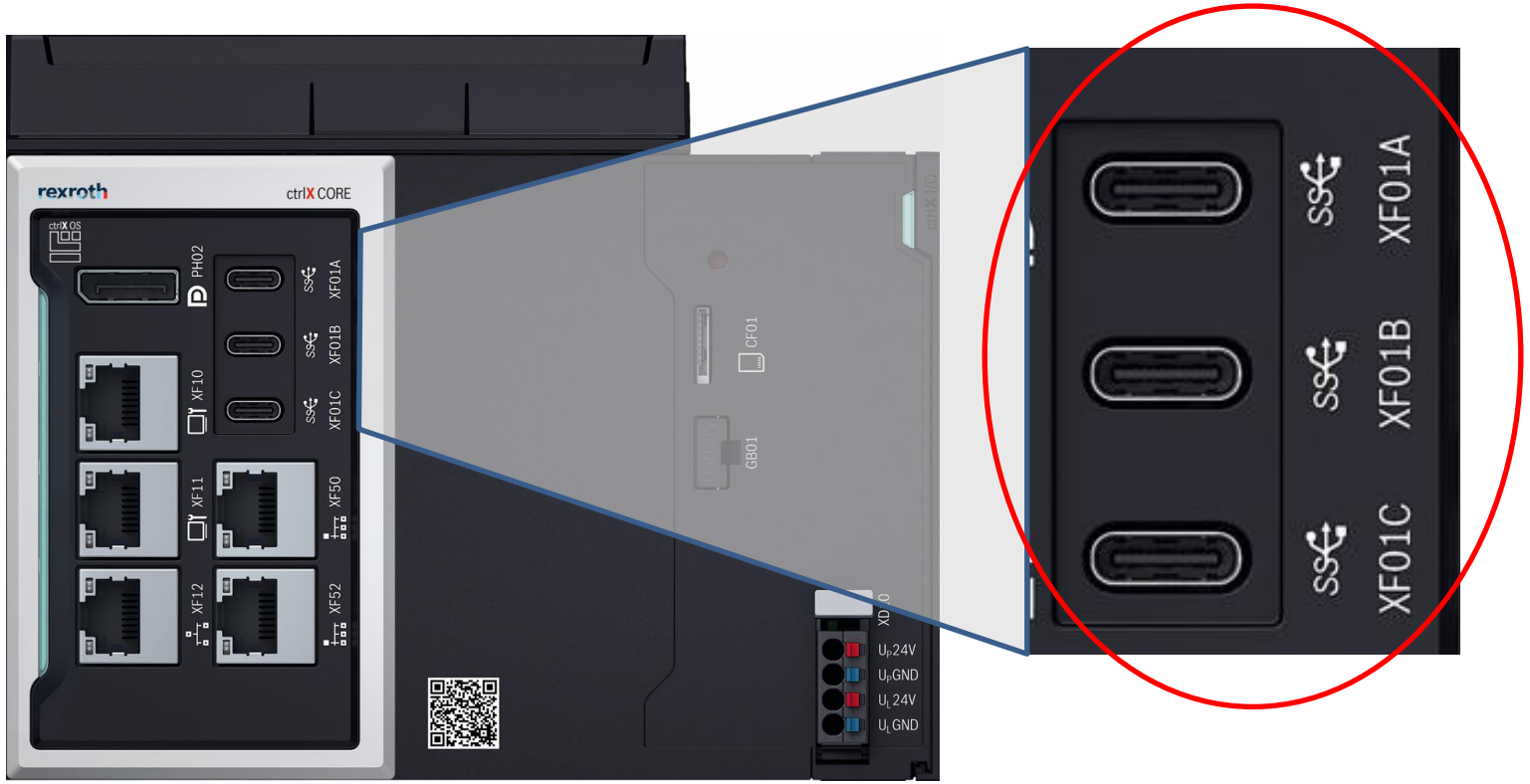
rexroth
A Bosch Company

- In the case of a ctrlX Core X7 that has three possible USB inputs, access to the USB inputs must be managed in a different way:

ctrlX CORE

192.168.1.5 | ctrlX-CORE Apply

Device name	ctrlX-CORE
System status	OPERATING
Hardware platform	amd64 (x64)
Serial number	7261404286740
Type code	COREX-M-X7-31-BNNN-21.01-01RS-NN-NN
App	1.20.4
License(s)	Basic, Default, Advanced, Performance
Ports	HTTPS:443 (OK) PLC:11740 (OK)



rexroth A Bosch Company

- Configuration test on the PLC part of the Datalayer reading with three USB:

Sin USB

```
strValue[0] ^ 'SystemData' ;
strValue[1] ^ ??? ;
strValue[2] ^ ]0' ;
strValue[3] ^ ??? ;
```

1º

```
strValue[0] ^ '66B3-F7A3' ;
strValue[1] ^ 'SystemData' ;
strValue[2] ^ ??? ;
strValue[3] ^ ??? ;
```

USB XF01C

2º

```
strValue[0] ^ '66B3-F7A3' ;
strValue[1] ^ 'D013-4F3A' ;
strValue[2] ^ 'SystemData' ;
strValue[3] ^ ??? ;
```

USB XF01B

3º

```
strValue[0] ^ '66B3-F7A3' ;
strValue[1] ^ '6E61-8429' ;
strValue[2] ^ 'D013-4F3A' ;
strValue[3] ^ 'SystemData' ;
strValue[4] ^ ??? ;
```

USB XF01A

1º

```
strValue[0] ^ '66B3-F7A3' ;
strValue[1] ^ 'SystemData' ;
strValue[2] ^ ??? ;
strValue[3] ^ ??? ;
```

USB XF01A

2º

```
strValue[0] ^ '66B3-F7A3' ;
strValue[1] ^ '6E61-8429' ;
strValue[2] ^ 'SystemData' ;
strValue[3] ^ ??? ;
```

USB XF01B

3º

```
strValue[0] ^ '66B3-F7A3' ;
strValue[1] ^ '6E61-8429' ;
strValue[2] ^ 'D013-4F3A' ;
strValue[3] ^ 'SystemData' ;
```

USB XF01C

1º

```
strValue[0] ^ '6E61-8429' ;
strValue[1] ^ 'SystemData' ;
strValue[2] ^ ??? ;
strValue[3] ^ ??? ;
strValue[4] ^ 'Data' ;
```

USB XF01B

2º

```
strValue[0] ^ '6E61-8429' ;
strValue[1] ^ 'D013-4F3A' ;
strValue[2] ^ 'SystemData' ;
strValue[3] ^ ??? ;
```

USB XF01A

3º

```
strValue[0] ^ '66B3-F7A3' ;
strValue[1] ^ '6E61-8429' ;
strValue[2] ^ 'D013-4F3A' ;
strValue[3] ^ 'SystemData' ;
```

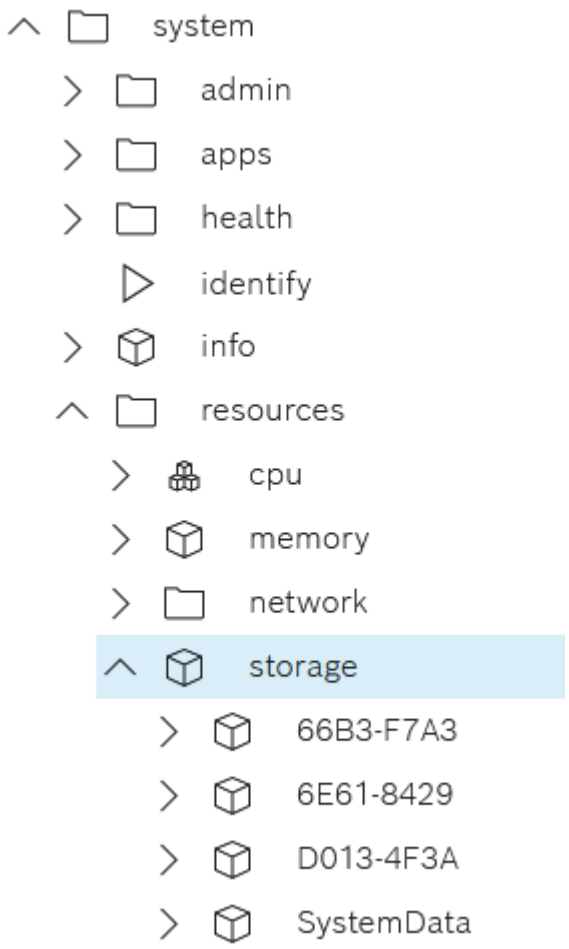
USB XF01C

The treatment, no matter what the USBs are put in, is always the same. In my opinion, the occupancy in the structure will depend on the "uuid" number, always from lowest to highest number in input Hexadecimal based on the first numbers of the USB.

26291 Dec	66B3	-	F7A3
28257 Dec	6E61	-	8429
53267 Dec	D013	-	4F3A

rexroth
A Bosch Company

- On the Datalayer side, the distribution is as follows:




```
{
  "storages": [
    {
      "size": 15711830016,
      "used": 107479040,
      "label": "TEST",
      "uuid": "D013-4F3A",
      "mounted": true,
      "format": "fat32",
      "device": "sdb1",
      "mountPoint": "/media/sdb1"
    },
    {
      "size": 7742750720,
      "used": 65536,
      "label": "STORE N GO",
      "uuid": "66B3-F7A3",
      "mounted": true,
      "format": "fat32",
      "device": "sdc1",
      "mountPoint": "/media/sdc1"
    },
    {
      "size": 15728607232,
      "used": 0,
      "label": "",
      "uuid": "6E61-8429",
      "mounted": false,
      "format": "fat32",
      "device": "sdd1",
      "mountPoint": ""
    },
    {
      "size": 37587709952,
      "used": 2949304320,
      "label": "ubuntu-data-enc",
      "uuid": "78cc6e9c-3278-4a0b-ad63-18b161f2ec58",
      "mounted": true,
      "format": "part",
      "device": "nvme0n1p7",
      "mountPoint": ""
    }
  ]
}
```


In this case, I recommend reading the datalayer of the "mountPoint" or "device" element to determine where we should assign the USB path. By default, the system, even without having "mounted" the usb, already assigns the path as can be seen in the image at the time of insertion:


```
1  {
2    "storages": [
3      {
4        "size": 15728607232,
5        "used": 0,
6        "label": "TEST",
7        "uuid": "D013-4F3A",
8        "mounted": false,
9        "format": "fat32",
10       "device": "sdb1",
11       "mountPoint": ""
12     },
13     {
```


Routes according to access:

/media/sdb1

/media/sdc1

/media/sdd1

Remember that the order in which the USBs are inserted will determine the Path assignment.

rexroth
A Bosch Company

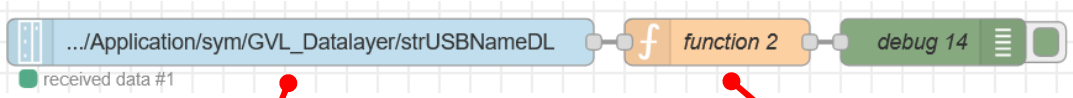
NodeRed for USB control

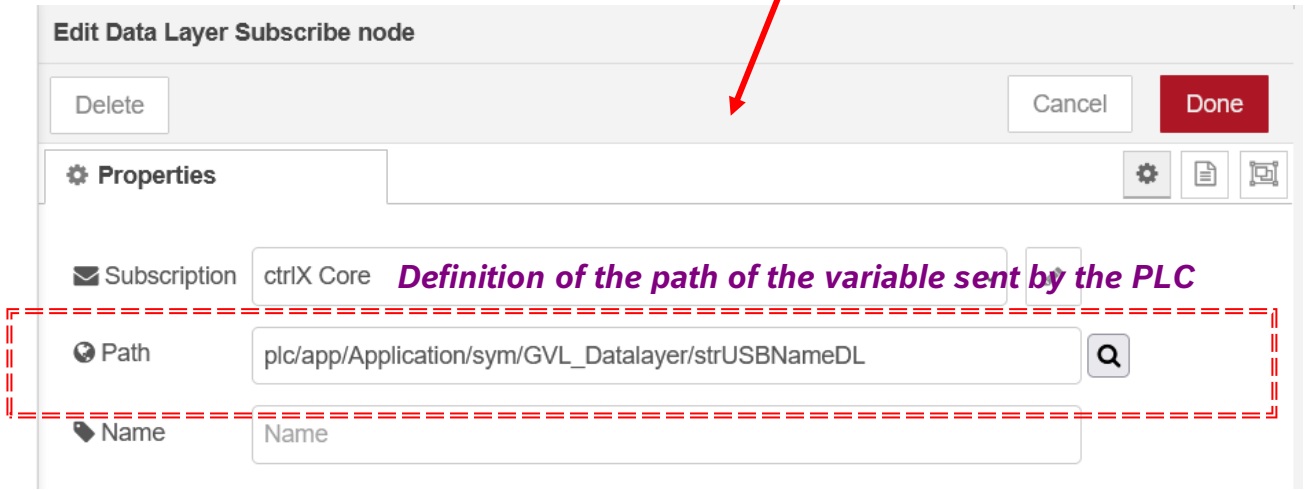
Node-RED for ctrlX

Some of the elements marked with ✸ have been used from the base created by our colleague Mauro Riboni

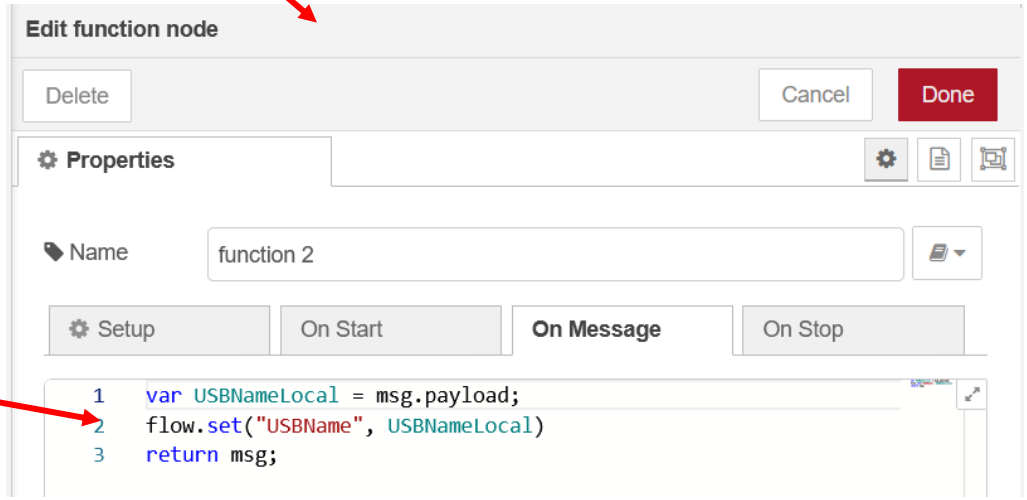
- *NodeRed used for USB name control:*

Identifier "uudi" of the Inserted USB





Definition of the path of the variable sent by the PLC



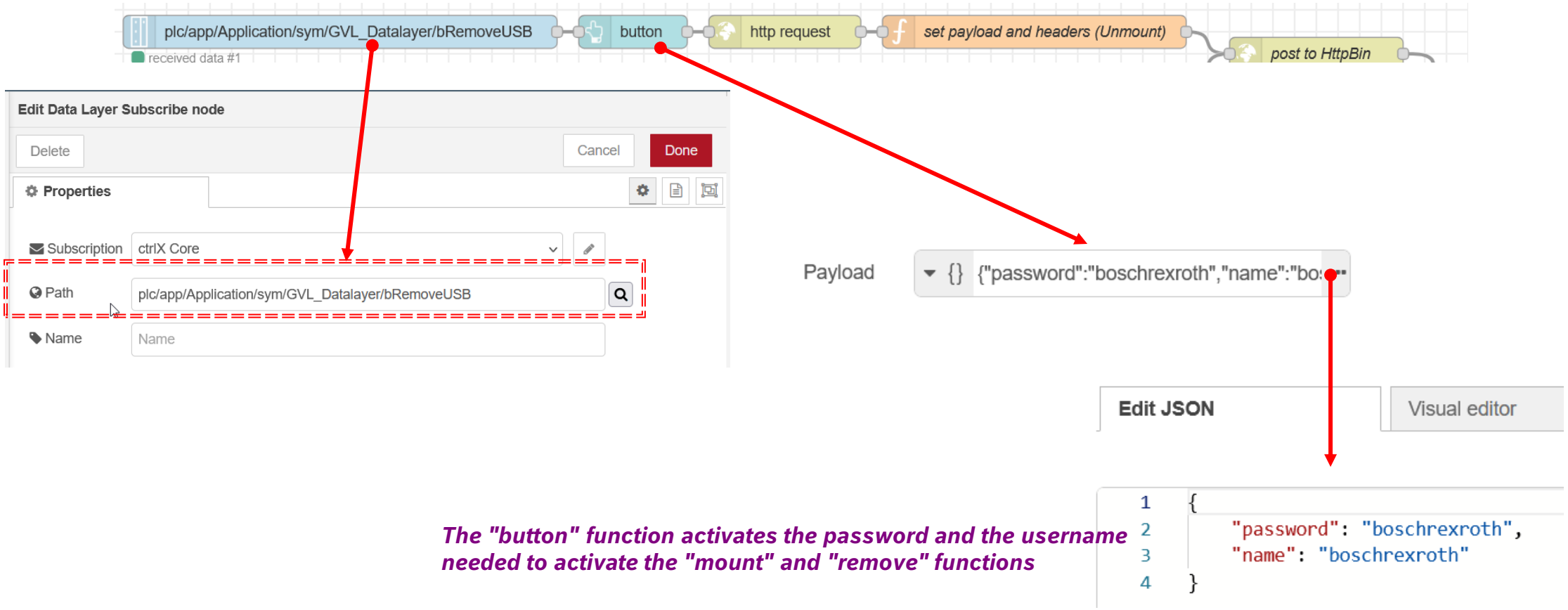
With flow.set we declare an internal variable of the flow and we can access it from anywhere in the flow.

We use a flow variable to dynamically control the name of the USB and that we will use in the "mount" and "remove" instructions.

Some of the elements marked with have been used from the base created by our colleague Mauro Riboni

rexroth
A Bosch Company

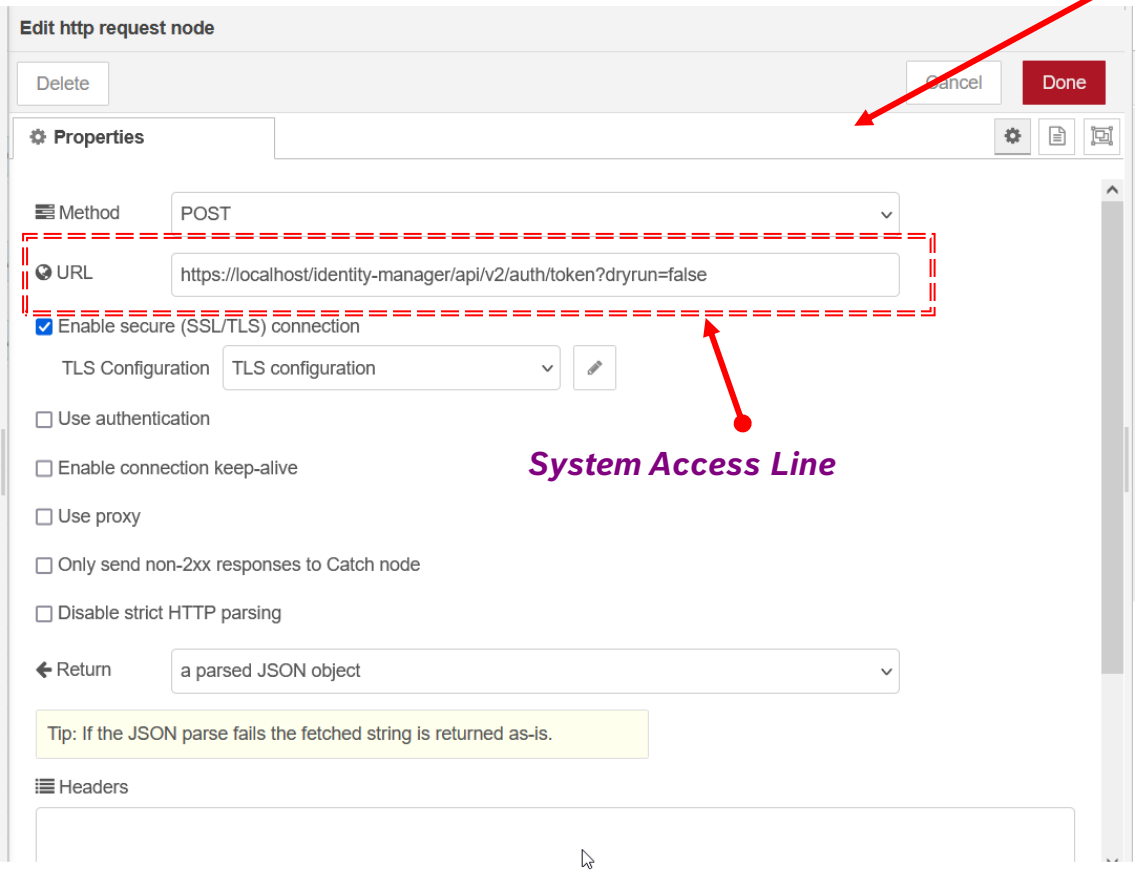
- *NodeRed used for "remove" control*



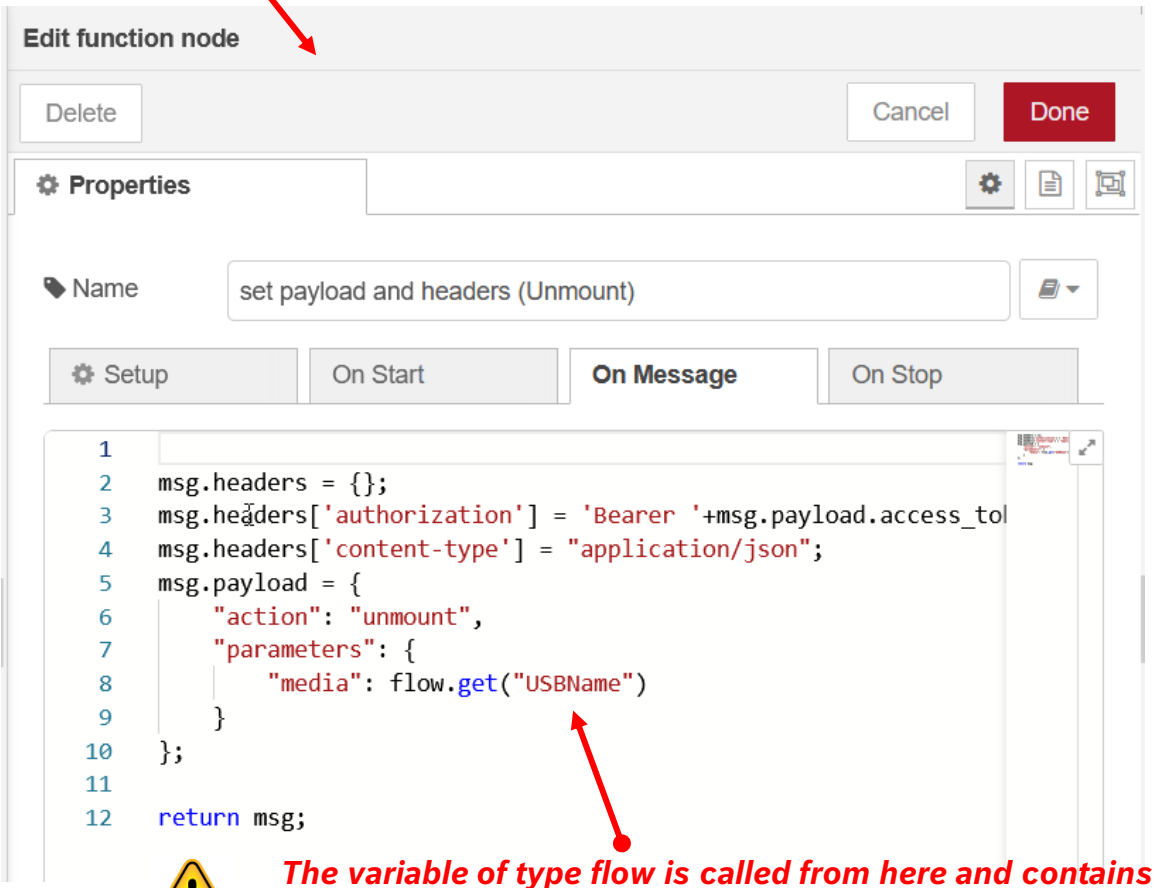
The "button" function activates the password and the username needed to activate the "mount" and "remove" functions

rexroth
A Bosch Company

- NodeRed used for "Remove" control



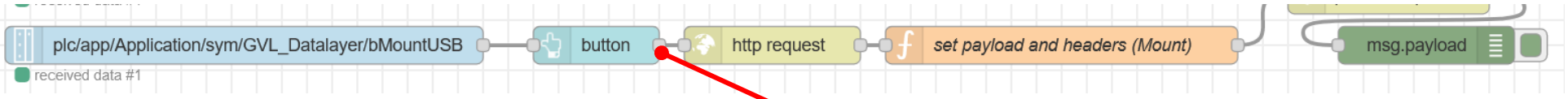
System Access Line

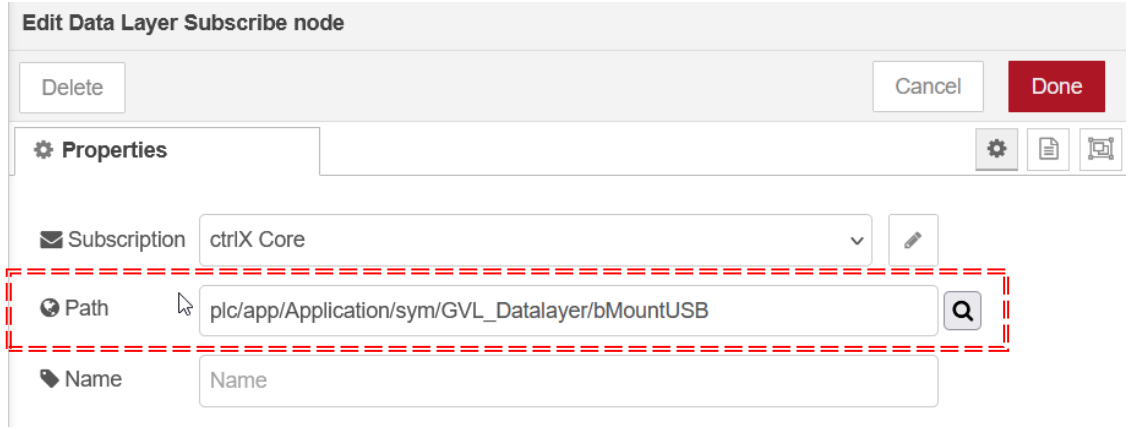


The variable of type flow is called from here and contains the "uudi" of the USB that we have connected

rexroth
A Bosch Company

- *NodeRed used for mount control*

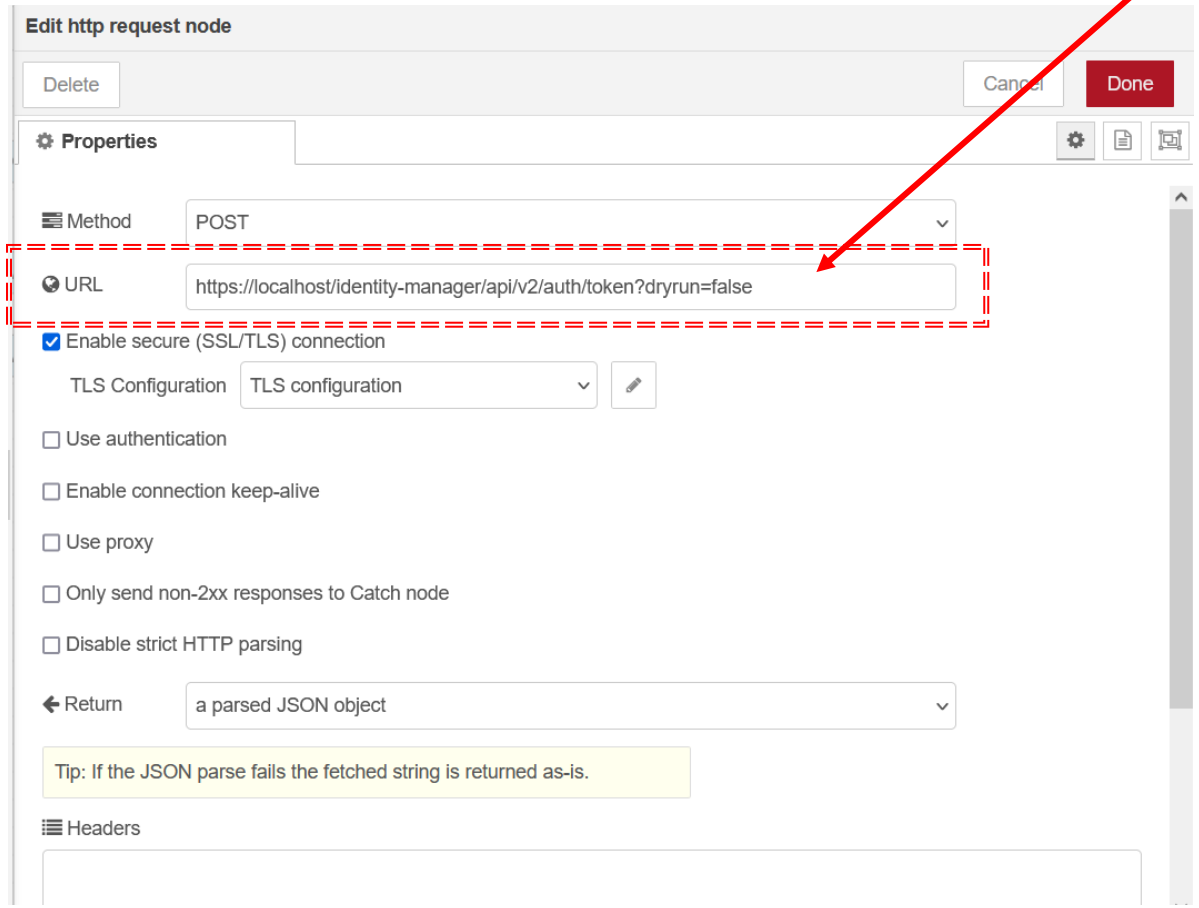


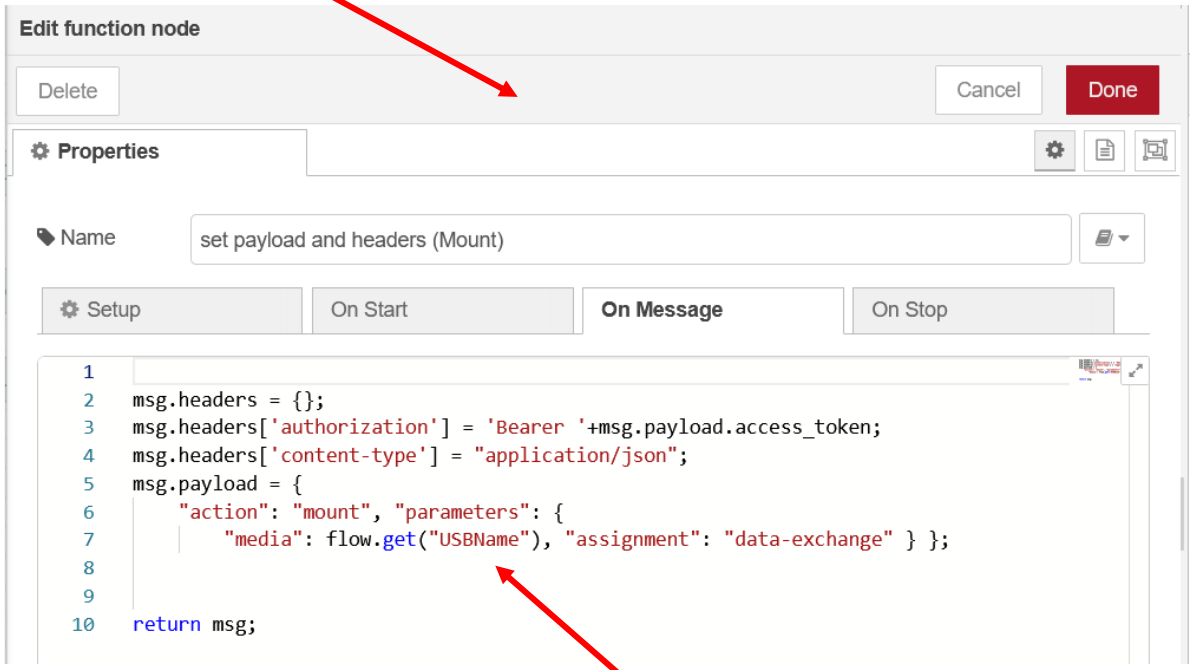


The "button" function activates the password and the username needed to activate the "mount" and "remove" functions

rexroth
A Bosch Company

- *NodeRed used for mount control*

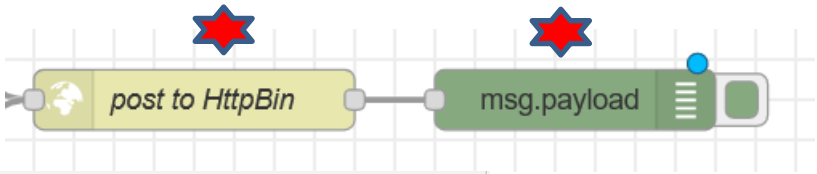


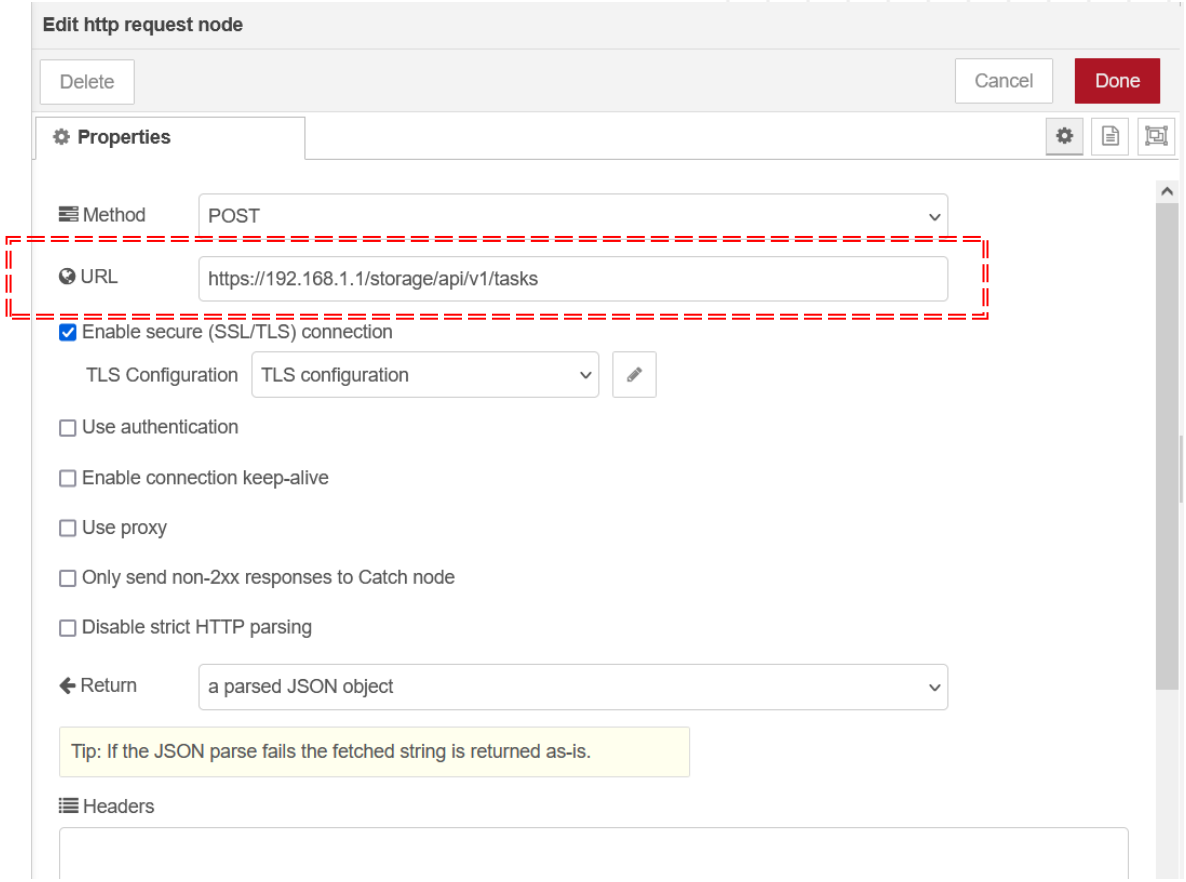


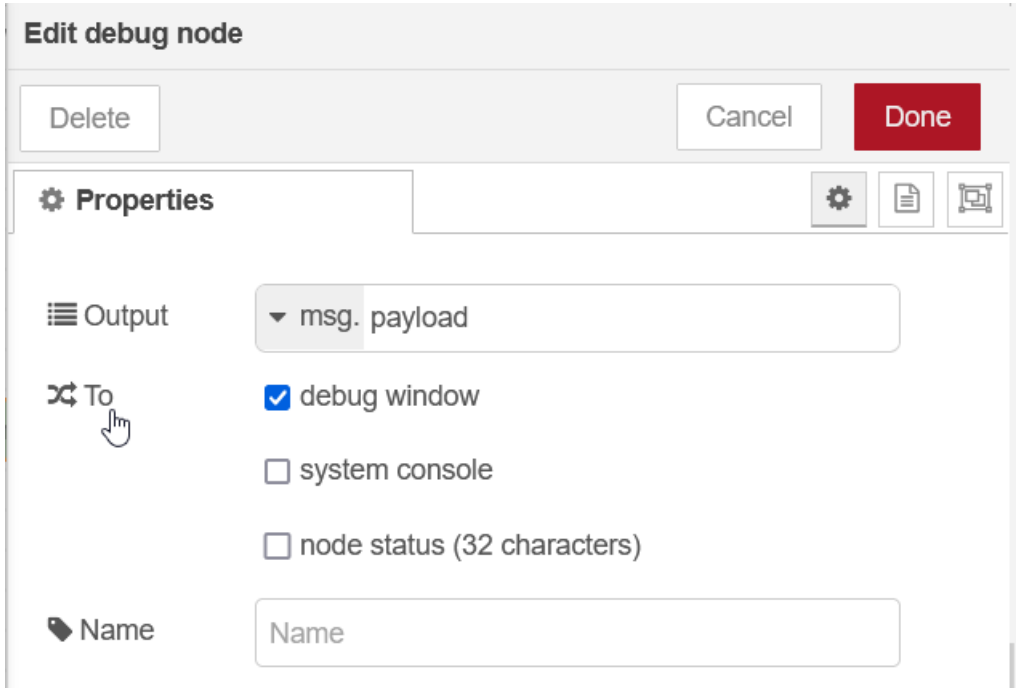
The variable of type flow is called from here and contains the "uudi" of the USB that we have connected

rexroth
A Bosch Company

- *NodeRed, common modules, used in mounting and removing*





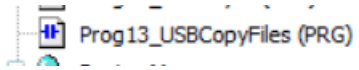


rexroth
A Bosch Company

Copy of the recipe directory to USB

ctrlX- *Copy of the recipe directory to USB*

- ***The last section that we are going to look at is the copy of the files, the complete directory on the USB for extraction, obviously it can also be managed the other way around and copy from the USB to the part of the recipes.***



Variables Used for System Control

```
PROGRAM Prog13_USBCopyFiles
VAR
// Control Bits from Screen
bCopyUSBToPLC:BOOL;
bCopyPLCToUSB:BOOL;

// Path USB And Recipes
strUSBPath: STRING :='/media/sda1/';
strPLCPath: STRING := 'TestRecipes/';

// Modules for Copy Control
fbDirCopyUSBToPLC: IL_DirCopyAsync;
fbDirCopyPLCToUSB: IL_DirCopyAsync;
END_VAR
```

Screen Buttons

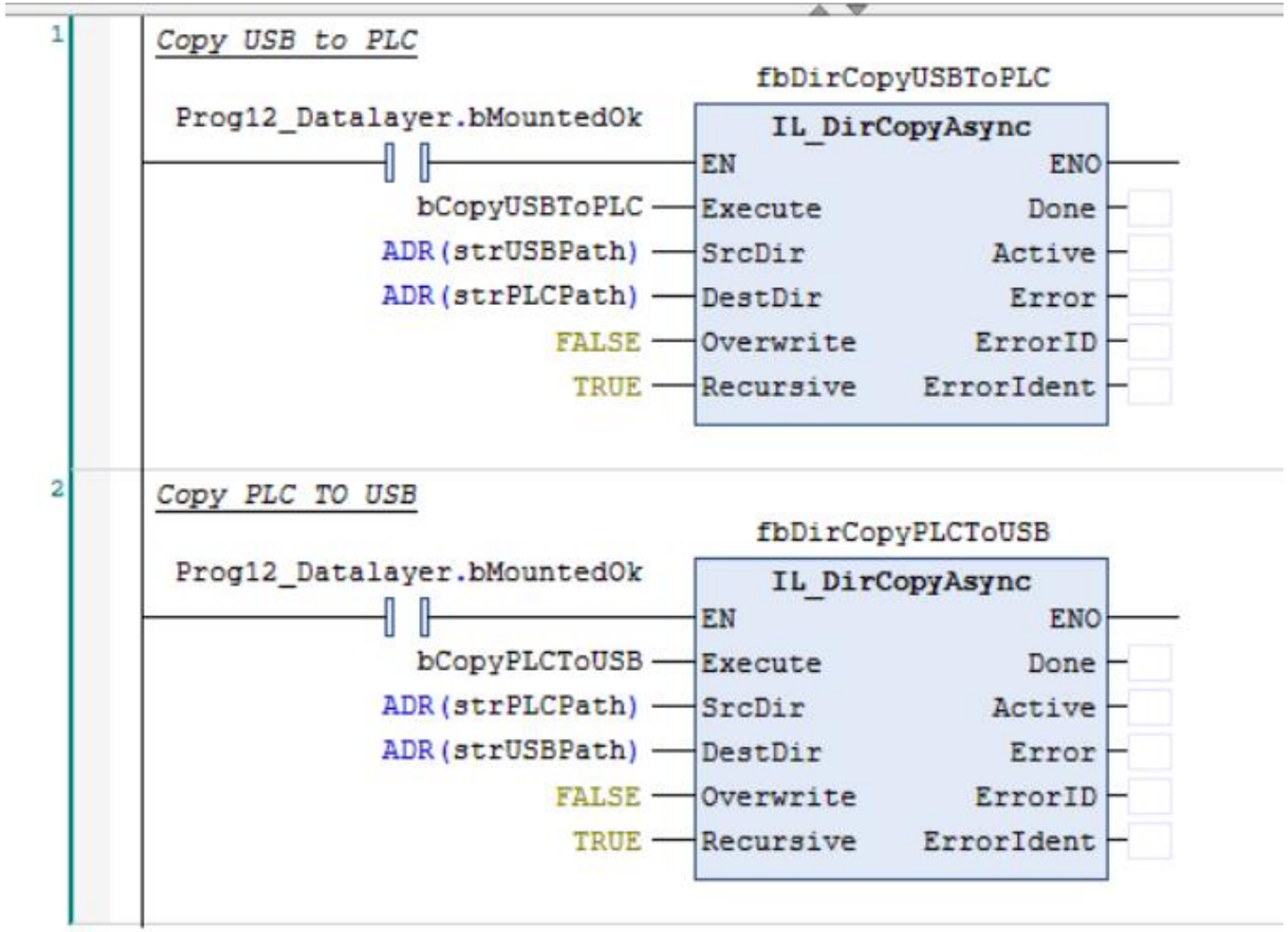
Fixed paths to the USB and the Recipes folder

Modules for copying directories

rexroth
A Bosch Company

- ***The program only has two modules and the copy is direct***

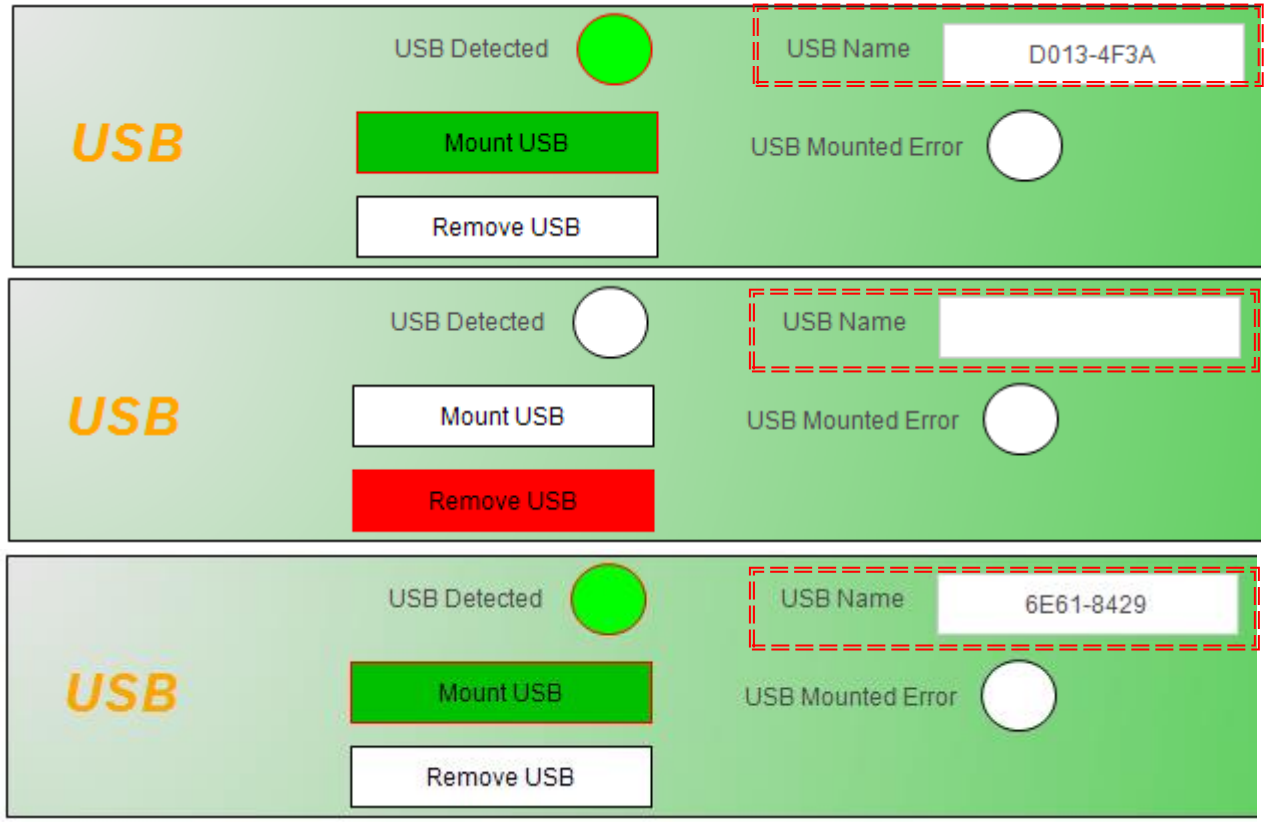
The only requirement to try to make the copy is that the USB is mounted



rexroth
A Bosch Company

- ***ADDITIONAL NOTES:***
- ***The "mount" depending on the type of USB takes a long time to manage (in some cases 3 minutes approx. or more)(ctrlX Core X3)***
- ***On some occasions it may be necessary to "reattach" the USB to restart the activation cycle***
- ***In some ctrlX Core with three USBs you should determine which one you want to work with, but in general the structure should be the same.***
- ***If necessary, you could modify the write paths and turn the system into something more dynamic***

Control screen view with different statuses (Two different USB and no USB)



USB Detected & Mounted

No USB

Another USB detected and mounted.

rexroth
A Bosch Company

Thank you for your attention

rexroth
A Bosch Company