Manual

FlexiVision

Schneider Electric Plug-In

ars automation

FlexiBowl® by ars

TABLE OF CONTENTS

1. Plug-In Installation
2. Plug-In Functions
    1. Recipe management functions
    2. Vision and movement functions
    3. Movement functions
3. FlexiVision® command List

This Plug-In was developed with the aim of facilitate communication between **Schneider robots (Scara, Delta 3Ax, Cartesian) and the FlexiVision® system**. With this integration, a fast and secure interface with the FlexiVision® vision system can be implemented simply.

The Plug-In provides a stable and immediate connection and the ability to send all the commands required for the correct operation of an application with a FlexiVision® vision system.

FlexiVision® Plug-In

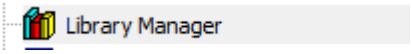
Schneider Electric

Plug-In Installation

Before you start working with the plug-in, make sure that the **'Machine Expert'** software is installed and configured correctly.
Export the **'FlexiVision.library'** file from the sent folder.

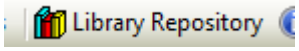
Step 1.

Enter the **'Library Manager'** section



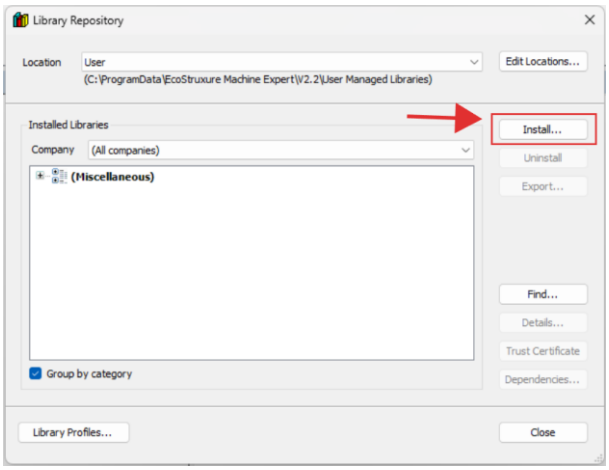
Step 2.

Enter the **'Library Repository'** section



Step 3.

Go to the **'Install...'** section and search for the **'FlexiVision.library'** library and install it.



Plug-In Functions

1. Communication function

1.1 FlexiVision – Connect

The Connect function allows the connection between FlexiVision and the robot by giving as input the IP address and open port in the PC where FlexiVision is open.

Function

`METHOD Connect : BOOL`

Input

`IP : STRING(16)` - IP of the PC on which the FlexiVision software is open.
`Port : UINT` - *FlexiVision port* opened in the 'Robot' section of the application.

Output values

`True` = The connection was successfully established.
`False` = An error occurred while connecting to FlexiVision.

1.2 FlexiVision - Disconnect

The Disconnect function allows you to disconnect the robot from the FlexiVision interface and clean the socket.

Function

`METHOD Disconnect : BOOL`

Output values

`True` = The connection was successfully established.
`False` = An error occurred while connecting to FlexiVision.

Plug-In Functions

1. Recipe management function

1.1 FlexiVision – Get Recipe

The Get_Recipe function allows the user to display the name of the recipe currently loaded in the FlexiVision interface.

The name of the recipe is saved in the variable designated '**q_sRecipeName**'.

Function

```
METHOD Get_Recipe : BOOL
```

Output values

`q_sRecipeName` = Name of the recipe uploaded to FlexiVision.

`q_xError` = True if an error occurred during the process, otherwise False.

`q_xDone` = True if the process is finished, otherwise False.

1.2 FlexiVision – Set Recipe

The Set_Recipe function allows the user to modify the recipe within the programme. Input the name of the recipe you want to upload to FlexiVision to load it.

Function

```
METHOD Set_Recipe : BOOL
```

Input

`STRING(64) i_sRecipeName` - The name of the recipe you wish to load in the FlexiVision interface.

Output values

`q_xError` = True if an error occurred during the process, otherwise False.

`q_xDone` = True if the process is finished, otherwise False.

Plug-In Functions

2. Vision and movement functions

2.1 FlexiVision – Start Empty

The Start_Empty function initiates a cleaning protocol by opening a small flap on the FlexiBowl and starting a rapid rotation sequence. This automatic process efficiently cleans the FlexiBowl of all objects, eliminating the need for operator intervention.

Function

```
METHOD Start_Empty : BOOL
```

Output values

`q_xError` = True if an error occurred during the process, otherwise False.

`q_xDone` = True if the process is finished, otherwise False.

`q_sResponseFlexiVision` = This variable contains the response of FlexiVIsion.

2.2 FlexiVision – Start Control

The Start Control function starts the activation procedure for the control camera. The response is customisable and is therefore saved in a variable to implement appropriate handling.

Function

```
METHOD Start_Control : BOOL
```

Output values

`q_xError` = True if an error occurred during the process, otherwise False.

`q_xDone` = True if the process is finished, otherwise False.

`q_sControlResponse` = This variable contains the response of FlexiVIsion.

2.3 FlexiVision – Start Locator

The Start_Locator function moves the FlexiBowl using the saved sequence until the object is detected and then returns its co-ordinates and sets them to Hopper_Flag, if necessary.

Function

```
METHOD Start_Locator : BOOL
```

Output values

`q_xError` = True if an error occurred during the process, otherwise False.

`q_xDone` = True if the process is finished, otherwise False.

`q_xPattern` = If True an object was found, you can read the co-ordinates in '**q_sPattern**'.

`q_stPattern` = [iPatternNumber, lrX. lrY, lrZ] you can find the co-ordinates in this struct.

`q_stPattern` = [iChannel, tTime] read from this buffer if xPattern = FALSE to obtain the output to raise to activate the hopper and for how long.

Plug-In Functions

2.4 FlexiVision – Turn Locator

The Turn Locator function moves the FlexiBowl® using the saved sequence, then takes a picture and checks if there is an object the robot can pick up.

Function

```
METHOD Turn_Locator : BOOL
```

Output values

q_xError = True if an error occurred during the process, otherwise False.
q_xDone = True if the process is finished, otherwise False.
q_xPattern = If True an object was found, you can read the co-ordinates in **'q_sPattern'**.
q_stPattern = [iPatternNumber, lrX. lrY, lrZ] you can find the co-ordinates in this struct.
q_stPattern = [iChannel, tTime] read from this buffer if xPattern = FALSE to obtain the output to raise to activate the hopper and for how long.

2.5 FlexiVision – Stop Locator

The Stop Locator function instantly terminates the FlexiVision® object search process.

Function

```
METHOD Stop_Locator : BOOL
```

Output values

q_xError = True if an error occurred during the process, otherwise False.
q_xDone = True if the process is finished, otherwise False.

2.6 FlexiVision - Test Locator

The Test_Locator function takes a picture from the vision system and returns the co-ordinates of the object found; if there is at least one withdrawable object in the FlexiBowl. If there are no withdrawable objects, it returns Null;null;null;null.

Function

```
METHOD Test_Locator : BOOL
```

Output values

q_xError = True if an error occurred during the process, otherwise False.
q_xDone = True if the process is finished, otherwise False.
q_xPattern = If True an object was found, you can read the co-ordinates in **'q_sPattern'**.
q_stPattern = [iPatternNumber, lrX. lrY, lrZ] you can find the co-ordinates in this struct.
q_stPattern = [iChannel, tTime] read from this buffer if xPattern = FALSE to obtain the output to raise to activate the hopper and for how long.

FlexiVision Command List

To send the command to FlexiVision®, the value of the string "command" must be changed.

N_Mission	Command	Action
1	“start_Locator"	Starts the parts localisation process by recalling the FlexiBowl® handling routine in case there are no parts that can be picked up. ***Return:*** “Pattern1;x;y;r”.
2	“stop_Locator"	Stops the process of locating the object with the aid of FlexiBowl®.
3	“turn_Locator"	If no parts are picked up, by this command the operator can make the Flexibowl® rotate and the “start_Locator” routine start. ***Return:*** “Pattern1;x;y;r”.
4	“test_Locator"	Starts the process of locating the object without the aid of FlexiBowl®. ***Return:*** “Pattern1;x;y;r”.
5	“start_Control"	Starts the inspection cycle. ***Return:*** “Control1;x;y;r”.
6	“state_Locator"	Locator status diagnostics is shown: ***Return:*** “Locator is Running” “Locator is in Error” “Locator is not Running”.
7	“start_Empty"	Start the FlexiBowl® Quick-Emptying sequence. ***Return:*** “start_Empty ended”
8	“get_Recipe"	The name of the recipe currently loaded on FlexiVision® is shown. ***Return:*** “recipe name”.
9	“set_Recipe=recipe name"	The recipe corresponding to the sent "recipe name" is loaded.