Manual

FlexiVision

ABB PLUG-IN

ars | Feeding Industrial Robotics

INDEX

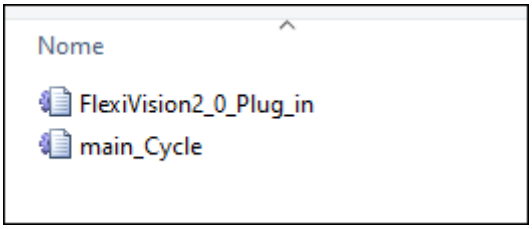
Questo Plugin è nato con l'idea di comunicare in maniera **rapida e sicura con il sistema Flexivision 2.0** tramite i robot **ABB,** mediante l' utilizzo di instruzioni in linuaggio **RAPID.**
Il Plugin necessita di due licenze aggiuntive per la gestione dei socket e del multitasking:
-Pc interface.
-Multi tasking
Per poter eseguire il server di **FlexiVision** in un task parallelo senza dover bloccare il ciclo principale del robot.

.

FlexiBowl® Plug-In

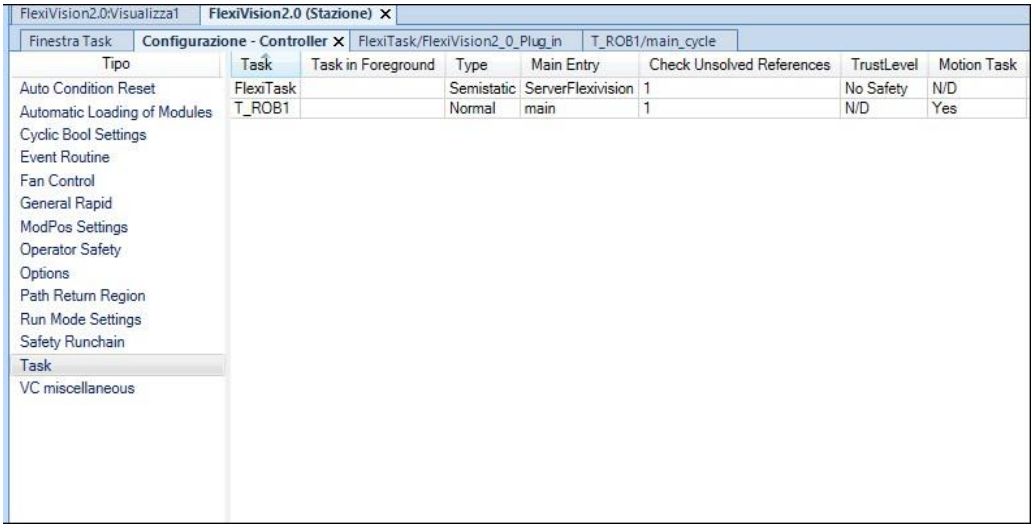
Plug-In Installation

Step 1.



Insieme a questa guida verrà fornito un esempio base sviluppato direttamente con RobotStudio, al fine di comprendere i passi per la realizzazione dell'applicazione.

Step 2.



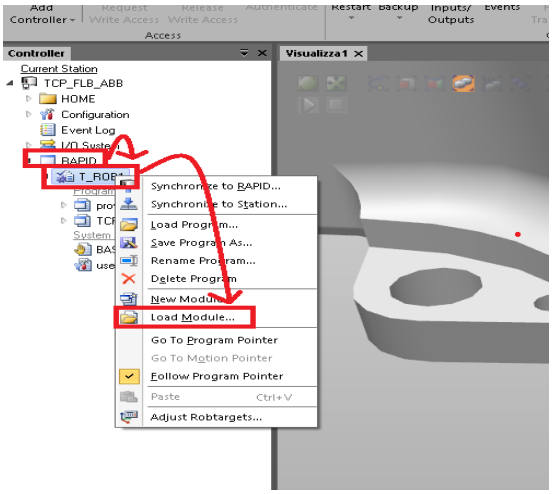
Dovremo utilizzare due task, uno per il ciclo principale ed uno parallelo che gestirà la comunicazione con **FlexiVision** senza mai essere fermato da stop o emergenze.

Da **Controller/Configuration/Controller/task** dopo aver richiesto l'accesso in scrittura sarà possibile aggiungere un nuovo task.

Creare un task di nome "**FLEXIVISION**".

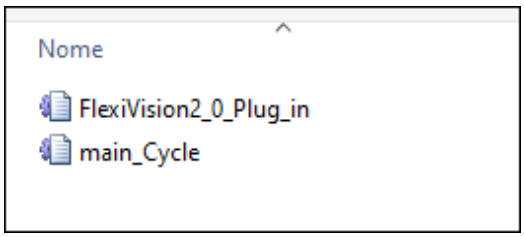
Plug-In Installation

Step 3.



Nel menù ***Controller*** selezionare: ***Rapid→T_ROB1*** Premere con il tasto destro su ***T_ROB1*** e selezionare ***Load Module***

Step 4.



Nella finestra di dialogo che apparirà , selezionare il Plugin "***main_Cycle"*** fornito da ARS.

Step 5.

Seguendo la stessa procedura caricare il modulo "***FlexiVision2_0_Plug_in***" nel task parallelo creato precedentemente e chiamato "**FLEXIVISION**".

Step 6.

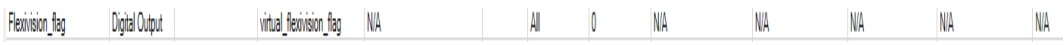
```
!assegnazione ip e porta al server
!bind ip & port of the server
SocketBind server_socket, "127.0.0.1", 4001;
```

Dopo aver importato il **Plug-in di FlexiVision**, ci basterà andare a modificare questa linea di codice andando ad impostare l' **IP** del nostro Robot e la porta che intendiamo utilizzare.

Plug-In Installation

Step 7.

Creare il segnale virtuale "**Flexivision_flag**". Tale segnale serve per raccordare le due task. Il percorso per la creazione del segnale è: **Controller/Configuration/I/O System/Signal.**
Il segnale da creare è riportato di seguito



Step 8.

Per inviare un comando a **FlexiVision** basterà inserire la stringa desiderata nella variabile "**command**".

FlexiVision Command List

To send the command to FlexiVision you must modify the value of the "command" string.

N_Mission	Command	Action
1	"start_Locator"	Starts the parts localization 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 the 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 the 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.

Script

Let's analyse the **"Flexivision2_0_Plug_in"** module.

```
MODULE FLEXIVISION

!Robtarget of the object to pick/control
    PERS robtarget pFlexi:=[[194,142,40],[6.12323E-
17,1,0,0],[0,0,0,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
    PERS robtarget  pControl:=[[0,0,0],[1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9]];

            !Flexivision string returned  PERS string
    return_from_flexivision;

            !Command to send to Flexivision  PERS string
    command;

            !Time activation Hopper  PERS num
    time;

            !Z value of the pFlexi robtarget. Set fixed according to object height  PERS num z_pos := 40;


PROC main()
            !Set semaphore Flexivision (doFlexivision) to 0.
            !Flexivision: 0 -> Robot turn | Flexivision: 1 -> Flexivision turn
            SetDO doFlexivision, 0;
    WaitTime 0.5;

            !Start Flexivision server
    ServerFlexivision;
ENDPROC


PROC ServerFlexivision()
        VAR socketdev server_socket;
        VAR socketstatus status;
        VAR socketdev client_socket;
        VAR string client_ip;
        VAR string receive_string;
        VAR string X;
        VAR string Y;
         VAR string Z;
        VAR string RX;
```

Script

```
VAR string partTmp;  VAR string
mission;  VAR string mission_case;
VAR num found;
VAR num found_start;
VAR num  found_end;  VAR num
found_len;  VAR num len;
VAR num anglez;  VAR num
x_pos;  VAR num y_pos;  VAR
num z_pos:=40;  VAR bool ok;

!Closing the previus socket instance
SocketClose server_socket;

!Creating new socket
SocketCreate server_socket;

!Bind ip & port of the server
SocketBind server_socket, "127.0.0.1", 4001;

!Start listening
SocketListen server_socket;

!Wait for a new connection
SocketAccept server_socket, client_socket\ClientAddress:=client_ip
\Time:=WAIT_MAX;


WHILE TRUE DO

  restart:
  !Wait until it's Flexivision turn (wait until doFlexivision is 1)
  WaitDO doFlexivision,1;
  mission := command;
  mission_case:= mission;
  IF(StrMatch(mission,1,"set_Recipe")<=StrLen(command)) THEN  mission_case:="set_Recipe=";
  ENDIF

restart_command:
  !Switch case according to the command to be sent
  TEST mission_case
```

Script

```
CASE "start_Locator":
    !Send the command according to the varible value
    SocketSend client_socket \Str := "start_Locator";

CASE "stop_Locator" :
    !Send the command according to the varible value
    SocketSend client_socket \Str := "stop_Locator"; GOTO
                restart;

CASE "turn_Locator" :
    !Send the command according to the varible value
    SocketSend client_socket \Str := "turn_Locator";

CASE "test_Locator" :
    !Send the command according to the varible value
    SocketSend client_socket \Str := "test_Locator";

CASE "start_Control" :
    !Send the command according to the varible value
    SocketSend client_socket \Str := "start_Control";

CASE "state_Locator" :
    !Send the command according to the varible value
    SocketSend client_socket \Str := "state_Locator";

CASE "start_Empty" :
    !Send the command according to the varible value
    SocketSend client_socket \Str := "start_Empty";

CASE "get_Recipe" :
    !Send the command according to the varible value
    SocketSend client_socket \Str := "get_Recipe";

CASE "set_Recipe=" :
    !Send the command according to the varible value
    SocketSend client_socket \Str := mission;
    GOTO        restart;

DEFAULT :
    TPWrite "undefined command"; GOTO
    restart;

ENDTEST
```

Script

```
!Receive the data
SocketReceive client_socket \Str :=
receive_string\Time:=WAIT_MAX;

!Memorize the lenght of the string received
 len := StrLen(receive_string);

!Global return from flexivision
 return_from_flexivision:=receive_string;

IF(StrMatch(receive_string,1,"Pattern")<=len) THEN
    !Once received the string, split the position and create the point

    !Pattern
    found_start:=0;
    found_end:=StrMatch(receive_string,found_start+1,";");  found_len:=found_end-found_start-1;
    NameModel :=  StrPart(receive_string,found_start+1,found_len);

    !X
    found_start:=StrMatch(receive_string,1,";");
    found_end:=StrMatch(receive_string,found_start+1,";");  found_len:=found_end-found_start-1;
    X := StrPart(receive_string,found_start+1,found_len);  ok:=StrToVal(X,x_pos);

    !Y
    partTmp:= StrPart(receive_string,found_end,len-found_end);  found_start:=StrMatch(partTmp,1,";");
    found_end:=StrMatch(partTmp,found_start+1,";");  found_len:=found_end-found_start-1;
    Y := StrPart(partTmp,found_start+1,found_len);  ok:=StrToVal(Y,y_pos);

    !RZ
    len := StrLen(partTmp);
    partTmp:= StrPart(partTmp,found_end,len-found_end+1);  found_start:=StrMatch(partTmp,1,";");
    found_end:=StrMatch(partTmp,found_start+1,";");  found_len:=found_end-found_start-1;
    RZ := StrPart(partTmp,found_start+1,found_len);  ok:=StrToVal(RZ,anglez);

    !Create the cartesion point
     pFlexi.trans:= [x_pos, y_pos, z_pos];
```

Script

```
!Define the rotation vector
    pFlexi.rot := OrientZYX(anglez, 0, 180);

    SetDO doFlexivision, 0;  ENDIF

IF(StrMatch(receive_string,1,"Control")<=len) THEN

    !Once received the string, split the position and create the point

    !Pattern
    found_start:=0;
    found_end:=StrMatch(receive_string,found_start+1,";");  found_len:=found_end-found_start-1;
    NameModel :=  StrPart(receive_string,found_start+1,found_len);

    !X
    found_start:=StrMatch(receive_string,1,";");
    found_end:=StrMatch(receive_string,found_start+1,";");  found_len:=found_end-found_start-1;
    X := StrPart(receive_string,found_start+1,found_len);  ok:=StrToVal(X,x_pos);

    !Y
    partTmp:= StrPart(receive_string,found_end,len-found_end);  found_start:=StrMatch(partTmp,1,";");
    found_end:=StrMatch(partTmp,found_start+1,";");  found_len:=found_end-found_start-1;
    Y := StrPart(partTmp,found_start+1,found_len);  ok:=StrToVal(Y,y_pos);

    !RZ
    len := StrLen(partTmp);
    partTmp:= StrPart(partTmp,found_end,len-found_end+1);  found_start:=StrMatch(partTmp,1,";");
    found_end:=StrMatch(partTmp,found_start+1,";");  found_len:=found_end-found_start-1;
    RZ := StrPart(partTmp,found_start+1,found_len);  ok:=StrToVal(RZ,anglez);

    !Create the cartesion point
    pControl.trans:= [x_pos, y_pos, z_pos];

    !Define the rotation vector
    pControl.rot := OrientZYX(anglez, 0, 180);
    SetDO doFlexivision, 0;
```

Script

```
IF(StrMatch(receive_string,1,"Hopper")<=len) THEN
                !Controll the hopper for a requested time without blocking the main
cycle
                found_start:=StrMatch(receive_string,8,";");
                found_end:=StrMatch(receive_string,found_start+1,";");
                found_len:=found_end-found_start;
                partTmp := StrPart(receive_string,found_start+1,found_len);  ok :=
                StrToVal(partTmp, time);

                HopperControll;
                GOTO restart_command;

            ENDIF

            GOTO
        restart;
        ENDWHILE

ERROR
    !In case of error:

    !Close the socket  SocketClose
    server_socket;

    !Restart the flexivision server
    ServerFlexivision;

ENDPROC

 PROC HopperControll()
    !Start the hopper for (time/1000) seconds whitout blocking the execution of  the program
    SetDO doHopper,1;
    SetDO \SDelay := time/1000, doHopper, 0;  ENDPROC

ENDMODULE
```

Script

Let's analyse the **"main_Cycle"** module.

```
MODULE MAIN_CYCLE
        !Robtarget of the object to pick/control
    PERS robtarget pFlexi :=[[194,142,40],[6.12323E-
17,1,0,0],[0,0,0,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];
    PERS robtarget pControl
:=[[0,0,0],[1,0,0,0],[0,0,0,0],[9E9,9E9,9E9,9E9,9E9,9E9,9E9]];

        !Robtarget of the Home and Place  PERS robtarget
    pHome :=[[27.64,-0.14,-
841.88],[0,1,0,0],[0,0,0,0],[9E+09,9E+09,9E+09,9E+09,9E+09,9E+09,9E+09]];
    PERS robtarget  pPlace:=[[-191.79,-79.74,-954.74],[0,-1,-5.52176E-38,-
2.76088E-38],[0,-1,0,0],[9E+9,9E+9,9E+9,9E+9,9E+9,9E+9,9E+9]];

        !Command to send to Flexivision  PERS string
    command;

        !Reference frame Flexivision  PERS wobjdata
    wobjFlexivision
:=[FALSE,TRUE,"",[[0.567452,0.062766,-
1000.11],[1,0,0,0.00000005]],[[0,0,0],[1,0,0,0]]];

        !Trigger data to trigger some IO when something happens  VAR triggdata trigFlexi;

        !Speed
    PERS speeddata vJoint:=[100,50,0,0];
        PERS speeddata vLinear:=[300,80,0,0];

    PROC main()
        SetDO doFlexivision,0;
        command:="start_Locator";

!Setup trigFlexi: when 150mm far from start set doFlexivision to 1.
!In this way, Flexivision start working even before the placing action.
        TriggIO trigFlexi, 150 \Start, \DOp:=doFlexivision, 1;

        WaitTime(0.5);
```

Script

Let's analyse the **"main_Cycle"** module.

```
!First action: go home.
  go_home;

!Now, at home, let Flexivision start  SetDO
doFlexivision,1;

!Pick and place continously
  WHILE TRUE DO
      pick_task;
      place_task
      ;
  ENDWHILE

ENDPROC


PROC go_home()
    MoveJ pHome,vJoint,fine,tool0;
ENDPROC

proc pick_task()
!Wait for Flexivision to finish the detection of the object. Flexivision=0  means Flexivision finished.
    WaitDO doFlexivision,0;

 !Go above the picking point and activate doSucker  MoveJDO Offs
(pFlexi,0,0,30),vJoint,z30,tool1\WObj:=wobjFlexivision,doSucker,1;

 !Go to picking point linearly and go above again
    MoveL pFlexi,vLinear,fine,tool1\WObj:=wobjFlexivision;
    MoveL Offs (pFlexi,0,0,70),vLinear,z30,tool1\WObj:=wobjFlexivision;

ENDPROC
```

Script

```
!First action: go home.
  go_home;

!Now, at home, let Flexivision start
SetDO doFlexivision,1;

!Pick and place continously
  WHILE TRUE DO
     pick_task;
     place_task;
  ENDWHILE

ENDPROC


PROC go_home()
   MoveJ pHome,vJoint,fine,tool0;
ENDPROC

proc pick_task()
!Wait for Flexivision to finish the detection of the object. Flexivision=0  means
Flexivision finished.
   WaitDO doFlexivision,0;

 !Go above the picking point and activate doSucker
   MoveJDO Offs
(pFlexi,0,0,30),vJoint,z30,tool1\WObj:=wobjFlexivision,doSucker,1;

 !Go to picking point linearly and go above again
   MoveL pFlexi,vLinear,fine,tool1\WObj:=wobjFlexivision;
   MoveL Offs (pFlexi,0,0,70),vLinear,z30,tool1\WObj:=wobjFlexivision;

ENDPROC
```

Script

```
    PROC place_task()
                    !Go above the placing point using TriggJ to trigger trigFlexi
                    !In this way, we let Flexivision start again (doFlexivision=1)  as soon
as the robot
                    !is 150mm far from the current point (picking point)
        TriggJ Offs(pPlace,0,0,70), vJoint, trigFlexi, z30, tool1 \WObj:=wobj0;

                    !Go to placing point linearly and go above again  MoveLDO
        pPlace,vLinear,fine,tool1 \WObj:=wobj0,doSucker,0;  MoveL Offs
        (pPlace,0,0,20),vLinear,z30,tool1 \WObj:=wobj0;
    ENDPROC
ENDMODULE
```