

Manual



FlexiVision

SHIBAURA PLUG-IN



Feeding
Industrial
Robotics

TABLE OF CONTENTS

1. Plug-In Installation
2. Starting a programme
3. Script
4. FlexiVision Command List

This Plugin was developed with the idea of communicating **quickly and safely with FlexiVision** and the **Shibaura Machine**.

The Plugin does not require additional Shibaura licenses. This PlugIn was developed *in the TSAssist work environment*.

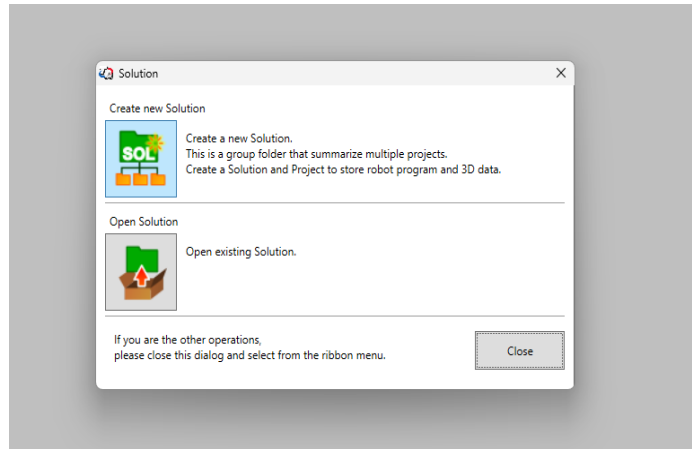
FlexiBowl[®] Plug-In

Shibaura Machine

View the Future with You

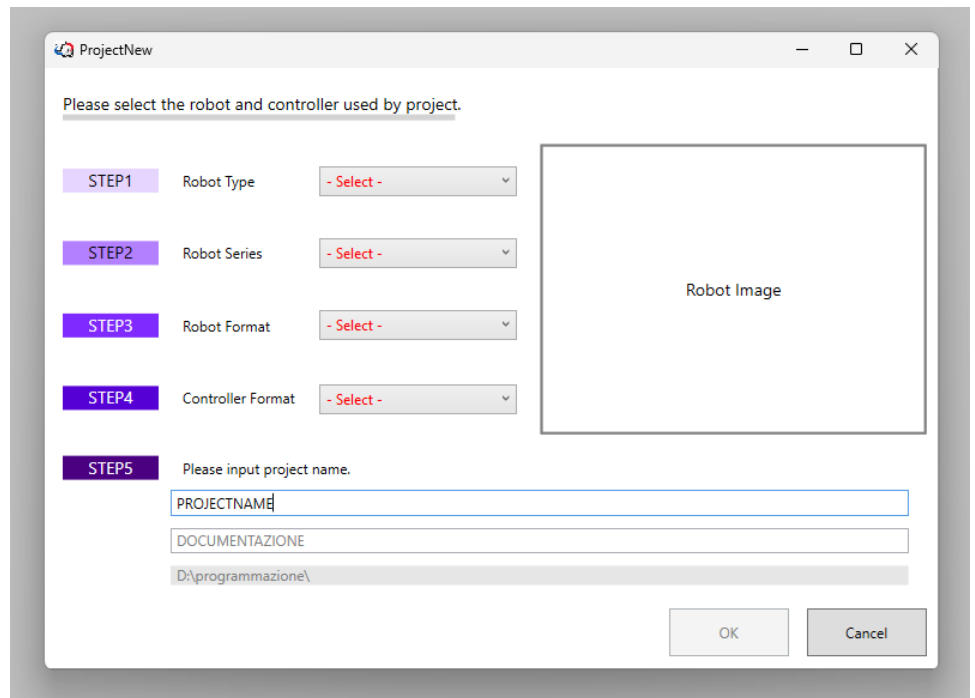
Plug-In Installation

Step 1.



Open TSAssist and click **"New Solution"**, enter the name and save path. Press OK. At the end of this procedure, an empty window is displayed.

Step 2.



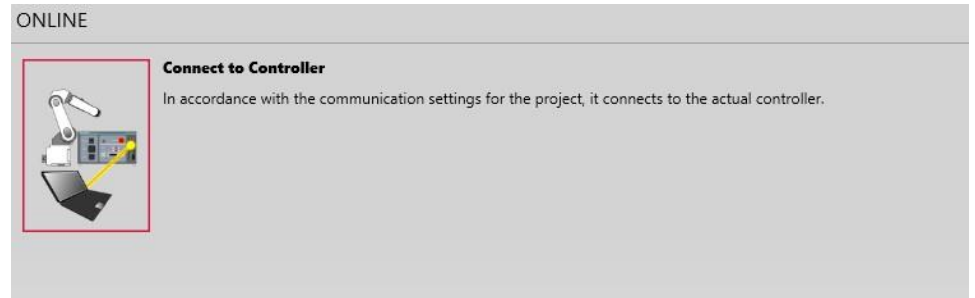
Now click on **new** in the top left-hand corner



then enter the required data from the window that opens and then using the different drop-down menus and after selecting all 5 fields click on **"OK"**

Plug-In Installation

Step 3.

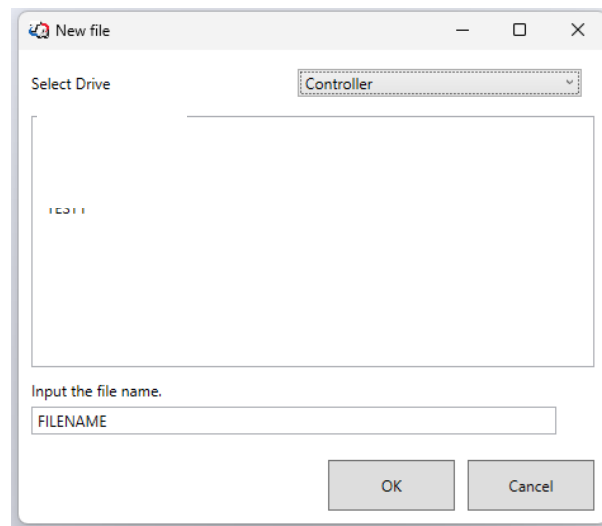


In the new window, click on Connect to Controller to connect to the controller.

Click on "**Online edit**" to open a mode for saving changes to the programme within the controller.



Create a new file by clicking on "**New**".

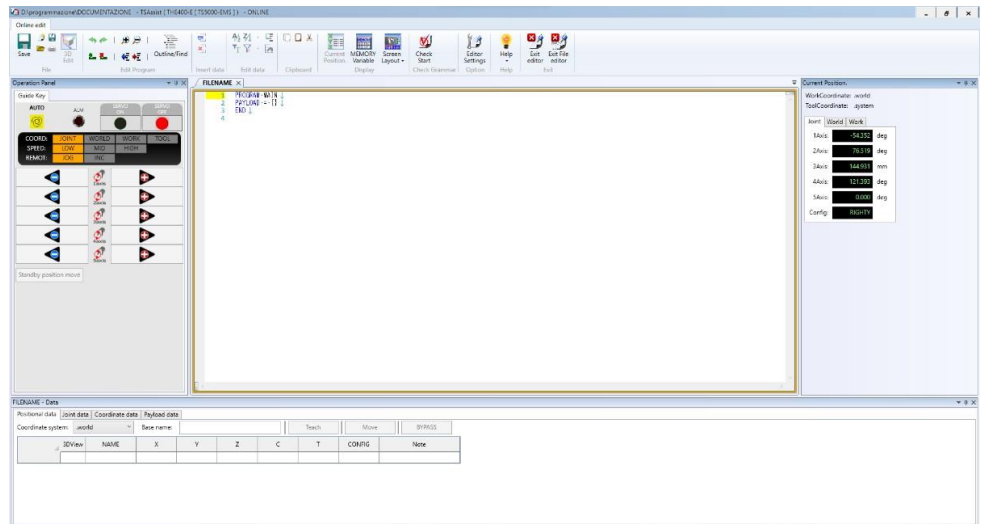


Enter the name you want to give the programme file and click "**OK**" to confirm the creation of the file.

Plug-In Installation

Step 4.

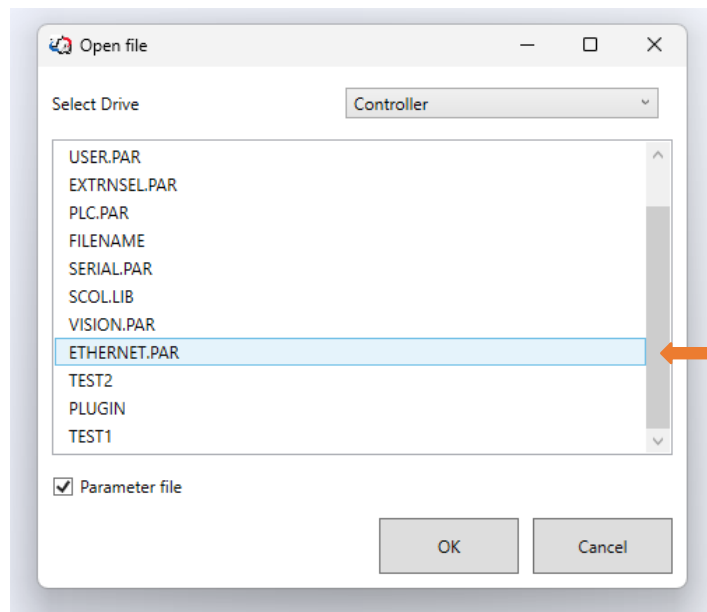
After clicking **"OK"** and clicking the file, the screen that appears is as follows.



To start the SetUp of the work environment, click on **"OPEN"** and click **"YES"**.



Select the file **"ETHERNET.PAR"** and then click on **"OK"**.



Plug-In Installation

Change the parameters [E000], [E001], [E002], [E0003], [E005], by entering the data you prefer

```

0
7 * [E000] · ROBOT · CONTROLLER · IP · ADDRESS ↓
8 IP-ADDRESS := · 192, · 168, · 1, · 20 ↓
9
10 * [E001] · SUBNET · MASK ↓
11 SUBNETMASK := · 255, · 255, · 255, · 0 ↓
12
13 * [E002] · DEFAULT · GATEWAY ↓
14 GATEWAY := · 192, · 168, · 1, · 1 ↓
15
16 * [E003] · ROBOT · COMMUNITY · NAME ↓
17 ROBO-COMMUNITY-NAME := · "TS5000" ↓
18
19 * [E004] · OPEN · MODE · IP1 · IP2 · IP3 · IP4 · IP5 · IP6 · IP7 · IP8 ↓
20 { 0:non-1:Robot-is-TCP-server-2:Robot-is-client-3:Robot-is-UDP } ↓
21 OPEN-MODE := · 1, · 0, · 0, · 0, · 0, · 0, · 0, · 0 ↓
22
23 * [E005] · OWN · PORT · NO ↓
24 { IP1 } ↓
25 OWN-PORT-NO[0] := · 10000 ↓
26 { IP2 } ↓
  
```

IP address of machine

Subnet Mask of machine

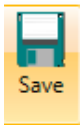
Default Gateway of machine

Name of machine

Address of port 1 of the machine (Strongly recommended 10000)

Step 5.

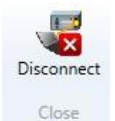
Click on **"Save"** after editing the parameters correctly on the previous page to save them within the controller.



Click on **"Exit File editor"** to leave the online editor.



Click on **"Disconnect"** to disconnect from the machine.

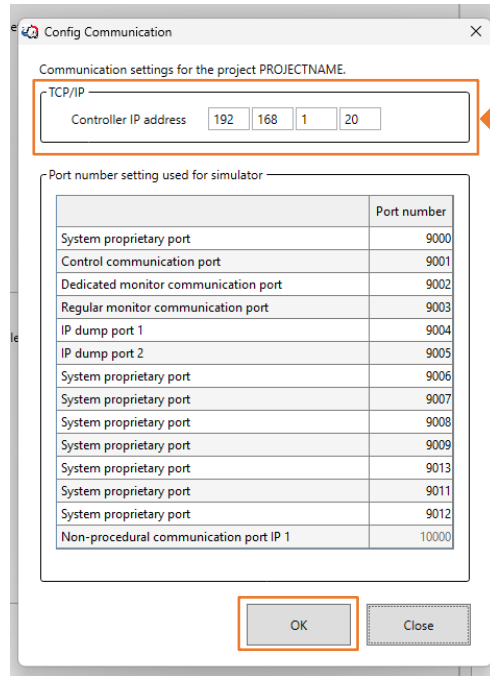


Click on **"Config connect info"** to go and correct the parameters we have just changed.



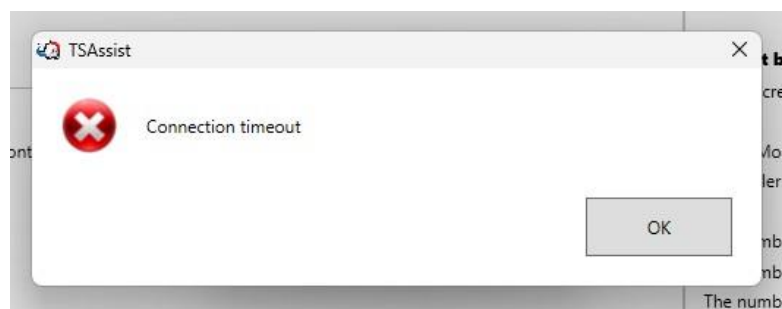
Plug-In Installation

Set "**Controller IP address**" to the value entered in the "**ETHERNET.PAR**" file and then click "**OK**"



Return to "**Online edit**" again by clicking on "**Connect to Controller**" and then on "**Online edit**" at the top left.

If trying to connect to the controller with the "**Connect to Controller**" button the IP address in "**Config Communication**" was entered incorrectly and TSAssist cannot find the robot, go back to "**Config Communication**" and enter the address correctly.

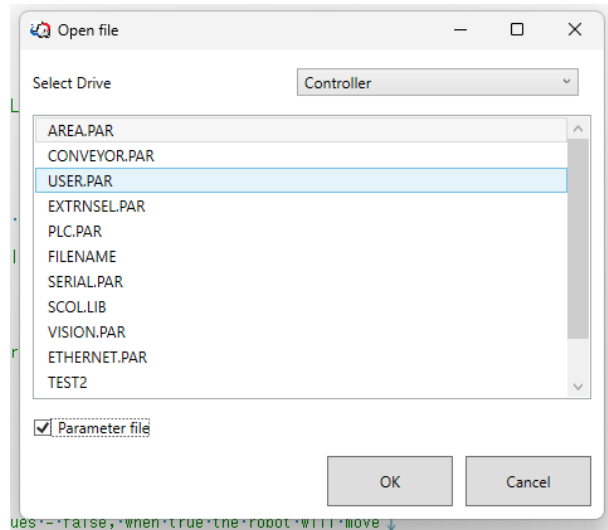


Plug-In Installation

To start the SetUp of the work environment, click on **"OPEN"** and click **"YES"**.



Select the file **"USER.PAR"** and then click on **"OK"**



In this file, scroll down to [U020] and [U021] and set

TESTRUN-SPEED = 50

SLOW-SPEED = 100

This sets the robot's MAXIMUM speed to 50% in **MANUAL mode** and 100% in **AUTO mode**.

```

130 ' [U020]-MANUAL-MODE-OVERRIDE-LIMIT-(1--100) [%] ↓
131 TESTRUN-SPEED=-50 ↓
132 ↓
133 ' [U021]-OVERRIDE-LIMIT-ON-SLOW-SPEED-SIGNAL-(1--100) [%] ↓
134 SLOW-SPEED=-100 ↓
    
```

Click on **"Save"** after correctly editing the parameters on the previous page to save them within the controller.



Plug-In Installation

Step 6.

Open TSAssist again and click on “**Open existing Solution**”.

Open Solution



Open existing Solution.

Now copy the project path into file explorer by adding “\PROJECTNAME\Working” where **PROJECTNAME** is the name of your previously decided project (example, in my case: D:\programming\DOCUMENTATION\PROJECTNAME\Working).

06/10/2023 16:27:59

D:\programmazione\DOCUMENTAZIONE

Find the previously created document where to insert the programme.

Nome	Ultima modifica	Tipo	Dimensione
AREA.PAR	27/07/2022 09:00	File PAR	20 KB
AUTOSTR	25/06/2021 14:01	File batch Windows	1 KB
CONVEYOR.PAR	27/07/2022 09:00	File PAR	11 KB
ETHERNET.PAR	27/07/2022 09:00	File PAR	2 KB
EXTRNSEL.PAR	27/07/2022 09:00	File PAR	17 KB
FILENAME	06/10/2023 15:23	File	1 KB
PLC.PAR	27/07/2022 09:00	File PAR	3 KB
SCOL.LIB	25/06/2021 14:01	Object File Library	2 KB
SERIAL.PAR	27/07/2022 09:00	File PAR	1 KB
TEST1	06/10/2023 16:05	File	6 KB
USER.PAR	06/10/2023 16:23	File PAR	7 KB
VISION.PAR	27/07/2022 09:00	File PAR	13 KB

Select the document and open it with “Notepad” or your preferred text editor (e.g. notePad++).

Delete the entire content from the text file until it is empty, and then copy the entire code reported below.

Starting a programme

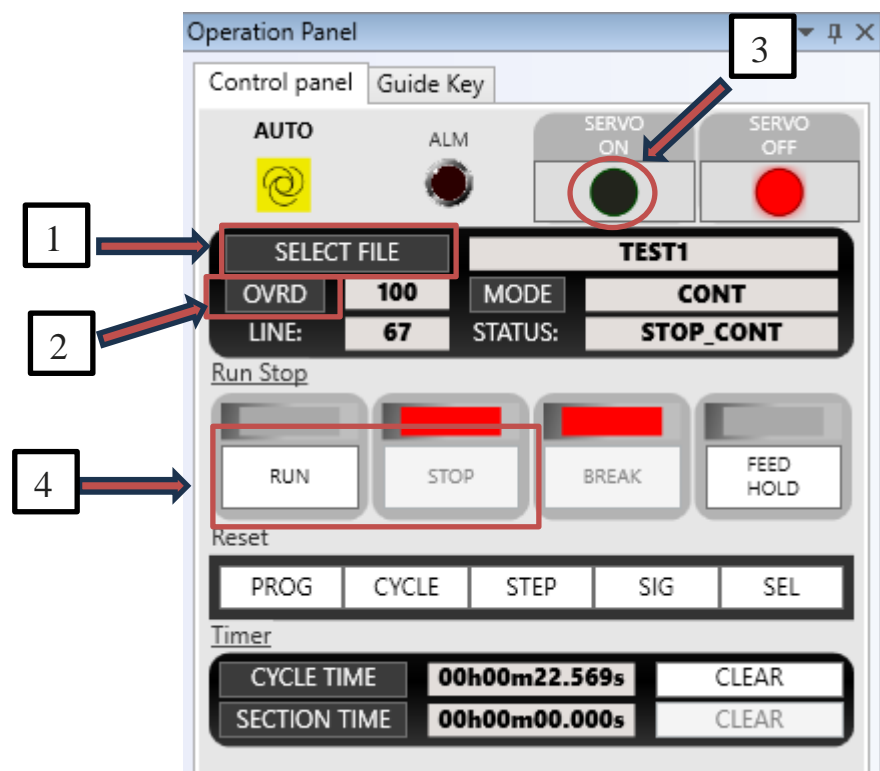
Step 1.

To start the programme manually, use the TP.

To start the programme in automatic mode (AUTO), turn the key on the TP towards **"AUTO"** and, if still in the online editor, click on **"Exit file editor"** at the top right



1. Select the programme file by pressing **"SELECT FILE"**
2. Select speed by pressing **"OVRD"**
3. Click on the grey circle below **"SERVO ON"** to activate it
4. Use the **"START"** and **"STOP"** buttons to start and stop the programme



Starting a programme

Step 2.

Trying to start the programme, however, one can see that, if the previous steps have been correctly replicated, the code launched by the robot will crash around line 57 at the line “**WAIT IP!STATUS == 5**”.

This is because FlexiVision has not been configured and is not connected to the robot.

After calibrating the camera and creating a model of an object, simply open the “**Robot**” section within the **Flexivision_RedPat** software.



Enter in the two textBoxes **IP** and **PORT** the IP address given to the robot and the robot port to which we want to connect, and then press “**Connection Test**”. If the robot is switched on and the data entered is correct, we will have the following visual feedback.



Script

GLOBAL

`DIM flexiVisionOut AS STRING(64) 'STRING INPUT VARIABLES`

`DIM x AS STRING(8)`

`DIM y AS STRING(8)`

`DIM r AS STRING(8)`

`DIM command as STRING(16)`

`DIM patternIndex as STRING(8)`

`DIM hopSign as STRING(8)`

`DIM hopTime as STRING(8)`

`DIM flexiError as STRING(1)`

`DIM recipeCheck as STRING(32)`

`DIM recipeName as STRING(32)`

`recipeCheck=""`

`DIM xNum as REAL 'NUMERIC VARIABLES`

`DIM yNum as REAL`

`DIM rNum as REAL`

`DIM yw as REAL`

`DIM pitch as REAL`

`DIM z as REAL`

`DIM hopTimeNum as REAL 'hopper time`

`DIM hopTimeSeconds as REAL`

`DIM hopSignNum as INT 'hopper signal number`

`DIM mainCount as INT`

`DIM robotSpeed as INT`

`DIM handSignal as INT`

`mainCount=0`

`DIM subID1 as INT 'variables to start the subtasks`

`DIM subID2 as INT`

`MAXTASK = 3`

`subID1 = 0`

`subID2 = 0`

`DIM moveFlag as INT`

`DIM hopperFlag as INT`

`DIM inputFlag as INT`

`moveFlag = 0 'BOOLEANS FLAGS`

`hopperFlag = 0 '1 = true, other values = false, when true the robot will move`

`inputFlag = 1`

`yw = 0.0 'CHANGE THIS VARIABLES IF NEEDED`

`(yw,pitch,z,dropPosition,recipeName,robotSpeed,handSignal)`

`pitch =0.0`

`z = 13 'z value of the obj`

`recipeName = "recipe_toshiba"`

`robotSpeed = 100`

`handSignal = 1`

`END`

Script

MAIN PROGRAM

```

IF mainCount == 0 THEN 'inizialization of the robot
  SPEED = robotSpeed
  MOVE HOME
  DOUT(-handSignal)
  TOOL = T1 'delate this line if no tools needed
  IF subID1 == 0 THEN subID1 = TASK("SUBINP")
  IF subID2 == 0 THEN subID2 = TASK("SUBHOPPER")
  PRINT TP, "tool setted: T1 ",CR
  IF IP1STATUS == 1 THEN IPOPEN(1)
  WAIT IP1STATUS == 5
  WHILE(NOT(recipeName == recipeCheck))
    PRINT IP1, "set_Recipe="+ recipeName,CR
    PRINT IP1, "get_Recipe",CR
    INPUT IP1, recipeCheck
    PRINT TP, "recipe setted: " + recipeCheck,CR
  ENDWHILE
  mainCount=1
ENDIF

PRINT TP, moveFlag,CR

IF moveFlag==1 THEN
  PRINT TP, "inp = 2",CR
  inputFlag = 2
  OBJ = OBJ + POINT(0,0,50)
  MOVE OBJ
  DOUT(handSignal)
  OBJ = OBJ - POINT(0,0,50)
  WAIT MOTION >= 80
  MOVE OBJ
  OBJ = OBJ + POINT(0,0,50)
  WAIT MOTION >= 80
  MOVE OBJ
  OBJ = OBJ - POINT(0,0,50)
  DROPPPOSITION = DROPPPOSITION + POINT(0,0,55)
  WAIT MOTION >= 80
  MOVE DROPPPOSITION
  WAIT MOTION >= 80
  moveFlag = 0
  inputFlag = 1
  PRINT TP, "inp = 1",CR
  DROPPPOSITION = DROPPPOSITION - POINT(0,0,55)
  MOVE DROPPPOSITION
  wait motion >= 100
  DOUT(-handSignal)
  DROPPPOSITION = DROPPPOSITION + POINT(0,0,55)
  WAIT MOTION >= 80
  MOVE DROPPPOSITION
  DROPPPOSITION = DROPPPOSITION - POINT(0,0,55)
ENDIF
END

```

Script

```

PROGRAM SUBINP
  WAIT inputFlag == 1
  PRINT IP1, "start_Locator",CR
  INPUT IP1, flexiVisionOut 'wants format "string + chr13" Hopper;3;1540
  PRINT TP, flexiVisionOut,CR
  flexiError = STRREAD(flexiVisionOut,1)

  IF flexiError == "#" THEN
    PRINT TP, "FLEXIVISION RUNNED IN AN ERROR: " + flexiVisionOut, CR
    moveFlag = 0
  ELSE
    flexiVisionOut = flexiError + flexiVisionOut
    command = STRSPLIT(flexiVisionOut, ";")
    IF command == "Hopper" THEN
      hopSign = STRSPLIT(flexiVisionOut, ";")
      hopTime = STRSPLIT(flexiVisionOut, ";")
      PRINT TP, "hopSign=" + hopSign,CR
      PRINT TP, "hopTime=" + hopTime,CR
      hopTimeNum = CREAL(hopTime)
      hopSignNum = CINT(hopSign)
      PRINT TP, hopSignNum,CR
      PRINT TP, hopTimeNum,CR
      hopTimeSeconds = hopTimeNum /1000
      PRINT TP, hopTimeSeconds,CR
      hopperFlag = 1 'can procede with the hopper activation
      inputFlag = 1
    ELSE
      command=STRSPLIT(command,"_")
      patternIndex = command
      x = STRSPLIT(flexiVisionOut, ";")
      y= STRSPLIT(flexiVisionOut, ";")
      r= flexiVisionOut
      xNum = CREAL(x)
      yNum = CREAL(y)
      rNum = CREAL(r)
      yNum = yNum * -1
      PRINT TP, "x=" + x,CR
      PRINT TP, xNum,CR
      PRINT TP, "y=" + y,CR
      PRINT TP, yNum,CR
      PRINT TP, "r=" + r,CR
      PRINT TP, rNum,CR
      OBJ = POINT(xNum,yNum,z,rNum,0)
      moveFlag = 1
      inputFlag = 2
    ENDIF
  ENDIF

  ENDIF
END

```

Script

```

PROGRAM SUBHOPPER
  IF hopperFlag == 1 THEN
    DOUT(hopSignNum)
    TIMER = hopTimeSeconds
    WAIT TIMER == 0
    DOUT(-hopSignNum)
    hopperFlag = 0
    inputFlag = 1
  ENDIF
END
DATA
TRANS T1          = 1.570, 0.291, 0.000, 0.000
TRANS WORK2       = 142.665, 237.566, 74.936, 143.566
WORK WORK2
POINT SAFEPOINT1  = 285.000, -210.000, 100.000, 0.000, 0.000 / FREE
POINT SAFEPOINT2  = 390.000, -210.000, 100.000, 0.000, 0.000 / FREE
POINT OBJ          = 0.000, 0.000, 0.000, 0.000, 0.000 / FREE
POINT DROPOSITION = -281.000, 159.000, 17.000, 0.000, 0.000 / FREE
POINT ORIGIN       = 0.000, 0.000, 20.000, 0.000, 0.000 / FREE
POINT ORIGINX      = 20.000, 0.000, 20.000, 0.000, 0.000 / FREE
POINT ORIGINY      = 0.000, 20.000, 20.000, 0.000, 0.000 / FREE
POINT HOME         = -330.000, 196.000, 70.000, 0.000, 0.000 / FREE

```


Now save the file from Notepad and open it from TSAssist to run it. In the case of changes to the code (apart from point values and variables), remember that placing comments not preceded by code may cause errors during compilation even if not detected by TSAssist.

Comment that has no compilation problems:

```
DIM xNum as REAL 'NUMERIC VARIABLES
```

Comment which may have compilation problems:

```
'NUMERIC VARIABLES
DIM xNum as REAL
```

Any changes to the code must be saved with the **"Save"**  on

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 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.