

ars

EPSON FLEXIBOWL PLUGIN



Questo Plugin è nato con l'idea di comunicare in maniera rapida e sicura con il FlexiBowl® tramite i robot Epson, mediante l'utilizzo di istruzioni in linguaggio RC+. Il Plugin non necessita di nessuna licenza aggiuntiva

FlexiBowl®

EPSON

SEIKO EPSON CORPORATION



EPSON RC+ 7.0

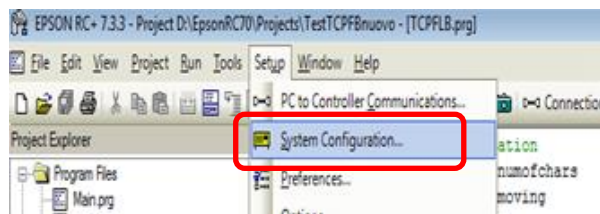
Version: 7.4.2

Copyright © 1994 - 2018

SEIKO EPSON CORPORATION

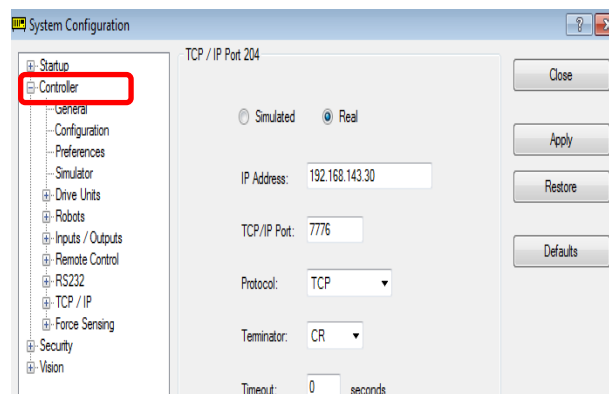


STEP 1:



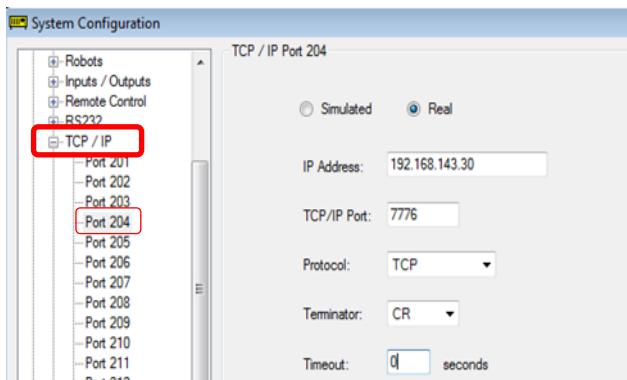
Nel menù Rc+ selezionare:
Setup → **System Configuration**

STEP 2:



Nella finestra di dialogo che apparirà ,
selezionare **Controller**

STEP 3:



Successivamente selezionare il menu
relativo alla comunicazione **TCP/IP**.
Una volta scelta la porta di
comunicazione ci basterà configurarla
seguendo il seguente standard

- **IP:** ip del flexibowl
- **Tcp/ip port:** 7776(standard)
- **Protocol:** TCP
- **Terminator:** CR

Epson RC+



STEP 4:

```
'opening com  
OpenNet #204 As Client  
WaitNet #204
```

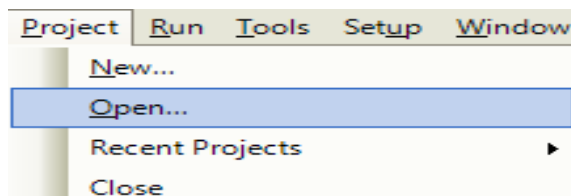
Il Plug-in Epson usa come standard la porta **TCP/IP 204**, se volessimo cambiarla ci basterà editare la parte di codice dove viene selezionata la porta da utilizzare .

STEP 5:



Copiare il Plug-in fornito da ARS nella cartella **projects** del programma **RC+**

STEP 6:



Aprire il Progetto del Plug-in precedentemente importato.

STEP 7:

Call TCPFLB(Command\$)

Basterà inserire questa linea di codice all' interno del nostro programma principale per poter comandare il **FlexiBowl®** in ogni sua funzione. Dovremo solo inserire il comando da eseguire come argomento.

COMMAND STRING FORMAT:

Sintassi corretta per ogni pacchetto			
Header		Command	Footer
Chr(0)	Chr(7)	Comando	Chr(13)

COMMAND LIST:

Comandi	Descrizione
QX2	Move
QX3	Move-Flip
QX4	Move-Flip-Blow
QX5	Move-Blow
QX6	Shake
QX7	Light on
QX8	Light off
QX9	Blow
QX10	Flip
QX11	Quick Emptying Option
QX12	Reset Alarm

SCRIPT:

```
1. 'variable declaration
2. Global Integer numofchars
3. Global Integer moving
4. Global Integer i
5. Global Integer pos
6. Global Double timeblow
7. Global String appoggio$
8. Global String appoggiomoving$
9. Global String data_tmp_Str$(0)
10. Global Byte inputbin(7)
11. Global Byte commandarray(5)
12. Global Byte movingbit(0)
13. Global Byte inputbyte
14. Global String messagearray$(0)
15.
16. Function TCPFLB(message$ As String)
17. ' Uncomment next line for debug mode
18. ' #define DEBUG 1
19.
20. 'manual test command ONLY IN DEBUG MODE
21. #ifdef DEBUG
22. Print "input command"
23. Line Input message$ 'Read one line input data into message
24. #endif
25.
26. 'clear vars
27. Redim commandarray(5)
28.
29. 'header
30. commandarray(0) = 0
31. commandarray(1) = 7
32.
33. 'command BODY
34. commandarray(2) = Asc("Q")
35. commandarray(3) = Asc("X")
36.
37. 'close communication char
38. commandarray(5) = 13
39.
40. 'opening com
41. OpenNet #204 As Client
42. WaitNet #204
43.
44. message$ = LCase$(message$)
45. 'select the correct type of movement according to the input
46. Select message$
```

SCRIPT:

```
47. '*****MOVE*****
48.     Case "move"
49.
50.     commandarray(4) = Asc("2")
51.
52.     #ifdef DEBUG
53.     Print "message =", message$
54.     Print "case = move"
55.     For i = 0 To UBound(commandarray)
56.     Print "pos", i, "=", commandarray(i)
57.     Next
58.     i = 0
59.     #endif
60.
61.     WriteBin #204, commandarray(), 6
62.     Call waitmoving
63.
64. '*****MOVE FLIP*****
65.     Case "move flip"
66.
67.     commandarray(4) = Asc("3")
68.
69.     #ifdef DEBUG
70.     Print "message =", message$
71.     Print "case = move flip"
72.     For i = 0 To UBound(commandarray)
73.     Print "pos", i, "=", commandarray(i)
74.     Next
75.     i = 0
76.     #endif
77.
78.     WriteBin #204, commandarray(), 6
79.     Call waitmoving
80. '*****MOVE FLIP BLOW*****
81.     Case "move blow flip"
82.
83.     commandarray(4) = Asc("4")
84.
85.     #ifdef DEBUG
86.     Print "message =", message$
87.     Print "case = move blow flip"
88.     For i = 0 To UBound(commandarray)
89.     Print "pos", i, "=", commandarray(i)
90.     Next
91.     i = 0
92.     #endif
93.
94.     WriteBin #204, commandarray(), 6
95.     Call waitmoving
96.
```

SCRIPT:

```

97. '*****MOVE BLOW*****
98.     Case "move blow"
99.
100.     commandarray(4) = Asc("5")
101.
102.     #ifdef DEBUG
103.     Print "message =", message$
104.     Print "case = move blow"
105.     For i = 0 To UBound(commandarray)
106.     Print "pos", i, "=", commandarray(i)
107.     Next
108.     i = 0
109.     #endif
110.
111.     WriteBin #204, commandarray(), 6
112.     Call waitmoving
113. '*****SHAKE*****
114.     Case "shake"
115.
116.     commandarray(4) = Asc("6")
117.
118.     #ifdef DEBUG
119.     Print "message =", message$
120.     Print "case = shake"
121.     For i = 0 To UBound(commandarray)
122.     Print "pos", i, "=", commandarray(i)
123.     Next
124.     i = 0
125.     #endif
126.
127.     WriteBin #204, commandarray(), 6
128.     Call waitmoving
129. '*****LIGNTON*****
130.     Case "lighton"
131.     commandarray(4) = Asc("7")
132.
133.     #ifdef DEBUG
134.     Print "message =", message$
135.     Print "case = shake"
136.     For i = 0 To UBound(commandarray)
137.     Print "pos", i, "=", commandarray(i)
138.     Next
139.     i = 0
140.     #endif
141.
142.     WriteBin #204, commandarray(), 6
143.
144.

```

SCRIPT:

```

145.  '*****LIGHTOFF*****
      *
146.      Case "lightoff"
147.
148.      commandarray(4) = Asc("8")
149.
150.      #ifdef DEBUG
151.      Print "message =", message$
152.      Print "case = shake"
153.      For i = 0 To UBound(commandarray)
154.      Print "pos", i, "=", commandarray(i)
155.      Next
156.      i = 0
157.      #endif
158.
159.      WriteBin #204, commandarray(), 6
160.
161.  '*****
162.      Default
163.  '*****BLOW*****
164.      If Not InStr(message$, "blow") Then
165.      'reallocate the command array
166.      Redim commandarray(7)
167.      'prepare the blowing request
168.      commandarray(0) = 0
169.      commandarray(1) = 7
170.      commandarray(2) = Asc("S")
171.      commandarray(3) = Asc("O")
172.      commandarray(4) = Asc("3")
173.      commandarray(6) = 13
174.      'extract the time for the blow
175.      pos = InStr(message$, "_")
176.      appoggio$ = Right$(message$, Len(message$) - pos)
177.      timeblow = Val(appoggio$)
178.
179.
180.      Print "blowing"
181.      commandarray(5) = Asc("L") 'S03L turn on blow
182.      WriteBin #204, commandarray(), 6
183.
184.      Wait timeblow
185.
186.      commandarray(5) = Asc("H") 'S03H turn off blow
187.      WriteBin #204, commandarray(), 6
188.
189.      Print "stop blowing"

```

SCRIPT:

```

190. '*****GENERIC COMMAND*****
191.     Else
192.     ' to upper message
193.     message$ = UCase$(message$)
194.     ' re allocate the dimension of the array according to the input string
195.     Redim commandarray(Len(message$) + 2)
196.     Redim messagearray$(Len(message$) - 1)
197.     commandarray(0) = 0
198.     commandarray(1) = 7
199.     appoggio$ = message$
200.     'split the input message in a chr array
201.     For i = 0 To (Len(message$) - 1)
202.         messagearray$(i) = Left$(appoggio$, 1)
203.         appoggio$ = Right$(appoggio$, Len(appoggio$) - 1)
204.     Next
205.     'format the commandarray
206.     For i = 0 To UBound(messagearray$)
207.         commandarray(i + 2) = (Asc(messagearray$(i)))
208.     Next
209.     commandarray(Len(message$) + 2) = 13
210.
211.     'send the command array
212.     WriteBin #204, commandarray(), UBound(commandarray) + 1
213.
214.     Call waitmoving
215.
216.     EndIf
217.
218.
219.
220.
221.
222.
223. '*****

```


SCRIPT:

```
224. Send
225.
226. CloseNet #204
227.
228. Fend
229. Function waitmoving
230. Print "Start..."
231. 'clear the commandarray
232. Redim commandarray(4)
233.
234. 'prepare the moving? request
235. commandarray(0) = 0
236. commandarray(1) = 7
237. commandarray(2) = Asc("S")
238. commandarray(3) = Asc("C")
239. commandarray(4) = 13
240.
241. 'wait the start of the movement
242. moving = 0
243. i = 0
244. numofchars = ChkNet(204)
245. If (numofchars >= 0) Then
246. ReadBin #204, inputbyte
247. EndIf
248. 'loop until the moving bit turn at 0 (chr48=0)
249. Do While (moving <> 48)
250.
251. WriteBin #204, commandarray(), 5
252. numofchars = ChkNet(204)
253. If (numofchars >= 5) Then
254. Redim inputbin(0)
255. Redim inputbin(numofchars)
256. ReadBin #204, inputbyte
257. ReadBin #204, inputbin(), numofchars
258. Redim movingbit(UBound(inputbin))
259. For i = 0 To UBound(inputbin)
260. If (inputbin(i) = 13 Or inputbin(i) = 37) Then
261. i = UBound(inputbin)
262. Else
263.
264. movingbit(i) = inputbin(i)
265.
266. EndIf
267. Next
268. moving = movingbit(5)
269. Wait 0.01
270.
271. EndIf
272.
273. Loop
274.
275. Print "Done..."
276.
277. Fend
```