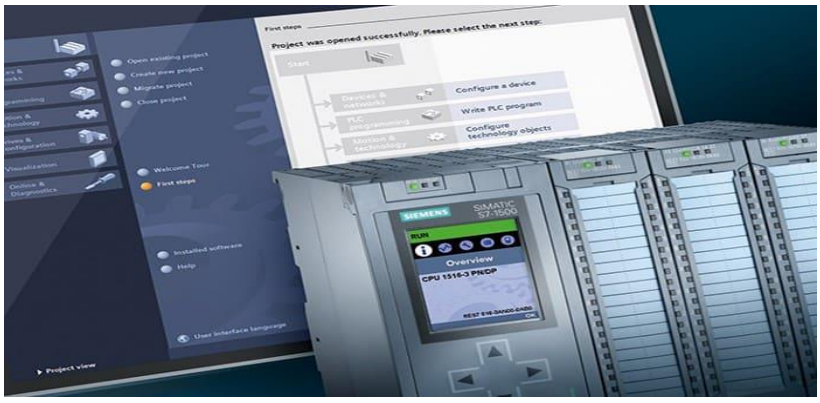
SIEMENS FLEXIBOWL PLUGIN

Questo Plugin è nato con l'idea di comunicare in maniera rapida e sicura con il FlexiBowl® tramite i PLC Siemens Serie 1200 e 1500, mediante l' utilizzo di istruzioni in linuaggio KOP/SCL.

Il Plugin sviluppato in TIA Portal V15 V15.1 V16 V17 non necessita di una licenza aggiuntiva ed è sviluppato per comunicare attraverso la scheda ethernet integrata dei PLC SIEMENS.

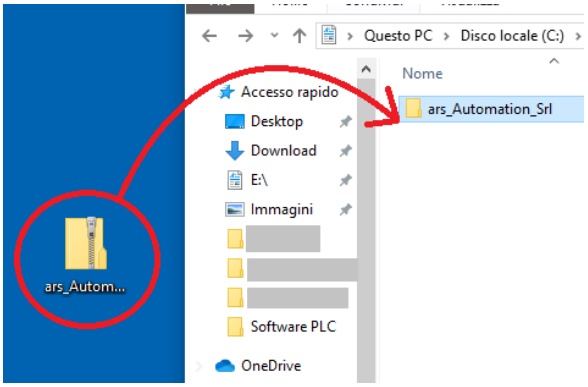
Richiede versione firmware 4.0 o superiore per PLC S7-1200
Richiede versione firmware 2.0 o superiore per PLC S7-1500
Richiede versione firmware 2.0 o superiore per PLC ET200SP

FlexiBowl®



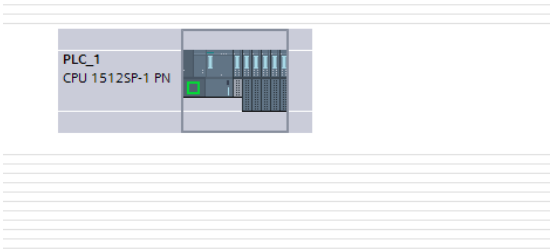
TIA Portal V15

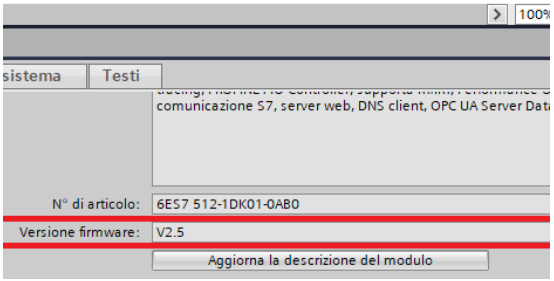
STEP 1:



Scompattare la Libreria “ars_Automation_Srl” in una cartella desiderata

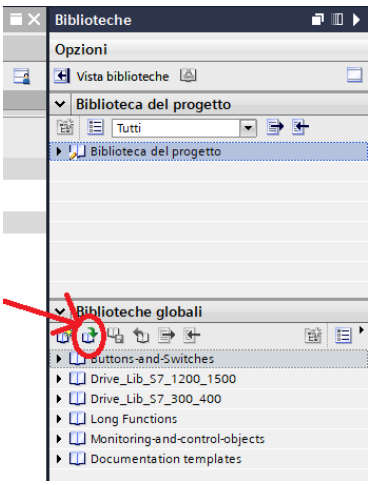
STEP 2:





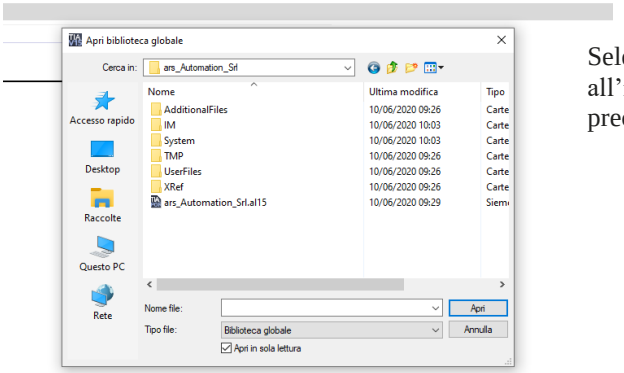
Accertarsi nella configurazione hardware del TIA Portal che la versione del firmware delle CPU sia maggiore o uguale alla 2.0 per i PLC S7-1500 o ET200 SP oppure maggiore o uguale alla 4.0 per i PLC S7-1200

STEP 3:



Cliccare nell’icona “Apri biblioteca globale” all’interno della sezione biblioteca in TIA portal.

STEP 4:



Selezionare quindi il file della libreria all'interno della cartella scompattata precedentemente.

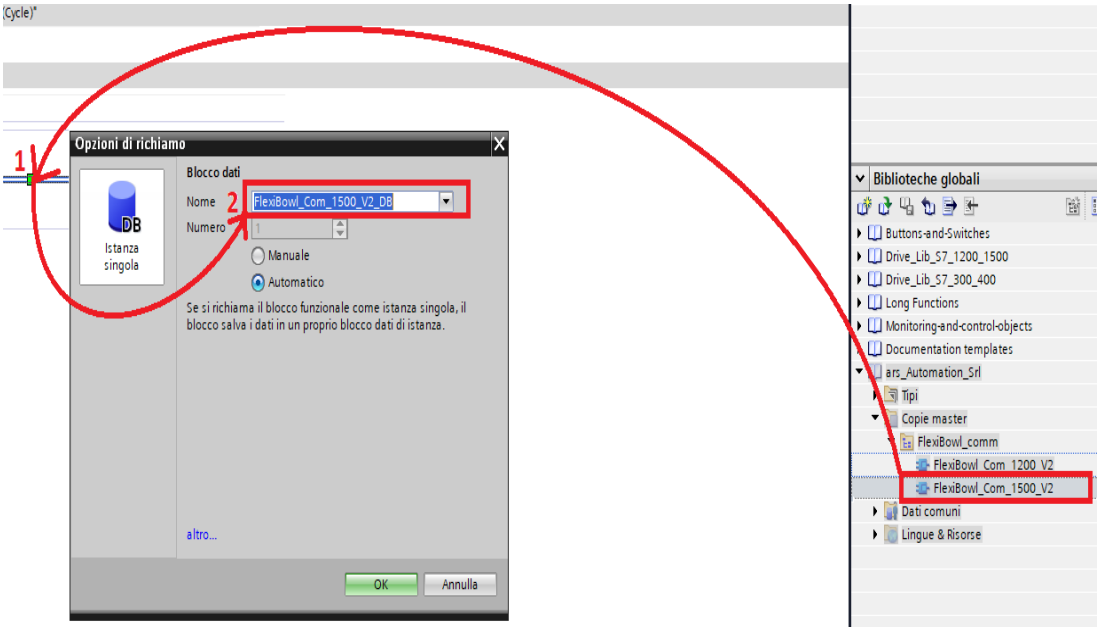
STEP 5:

Trascinare all'interno del software il blocco di comunicazione ora disponibile nella libreria; selezionare il blocco in base alla CPU che si sta utilizzando:

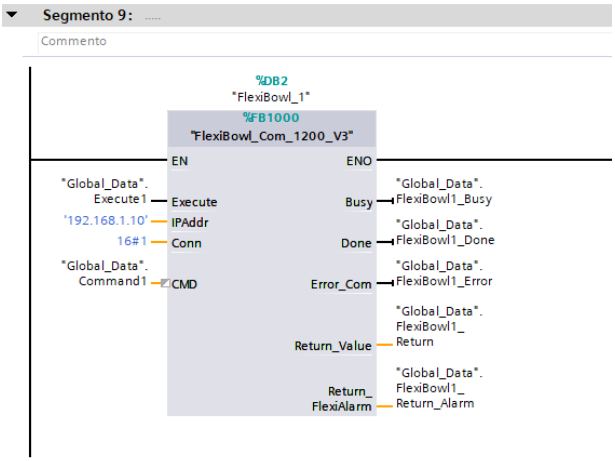
- **FlexiBowl_com_1200_V2 per S7-1200;**
- **FlexiBowl_com_1500_V2 per S7-1500 o ET200 SP;**

Una vola inserito il blocco all'interno del software verrà chiesto di dare un DB di istanza.
È possibile creare anche blocchi multi-istanza.

NB: Si consiglia di utilizzare un solo blocco per ogni FlexiBowl che si intende far comunicare e di gestire le richieste da inviare a monte del blocco, avvalendosi delle uscite "Done" e "Busy" del blocco



STEP 6:



Popolare ora gli ingressi e le uscite del blocco in base alle proprie esigenze:

TAG	Tipo	Descrizione
EN	IN:Bool	Abilitazione del blocco. Deve essere sempre abilitato poiché all'interno ci sono delle istruzioni che si occupano di tenere attiva la connessione TCP
Execute	IN:Bool	Invia il comando presente all'ingresso "CMD". Il blocco prende in considerazione questo ingresso solo per un ciclo di scansione del PLC e se la comunicazione è stabilita.
IPAddr	IN:String	Indirizzo IP del FlexiBowl. Specificare l'indirizzo IP del Flexibowl. La lungezza della stringa è vincolata, pertanto è necessario riempire tutte e 4 le triadi con tre numeri es: 192.168.0.161 → '192.168.000.161' 10.100.120.5 → '010.100.120.005' Le istruzioni TCP di comunicazione sono ad alto livello e gestite dal soket ethernet del PLC pertanto anche se il FlexiBowl si trova in un'altra sottorete ed è presente un gateway di riferimento impostato nella configurazione hardware del PLC è possibile specificare direttamente l'indirizzo IP finale del FlexiBowl.

STEP 6:

TAG	Tipo	Descrizione
CMD	IN:String	Comando in formato stringa da inviare al flexiBowl (vedere tabella più avanti)
Conn	IN: CONN_OUC	ID della connsessione, questo parametro deve essere passato in esadecimale es: “16#01” e deve essere un numero univoco compreso tra 1 ed il limite massimo di connessioni TCP che supporta il PLC es 32
ENO	OUT:Bool	Esecuzione del blocco andata a buon fine senza alcun errore software
Busy	OUT:Bool	Esecuzione QX in corso da parte del FlexiBowl, pertanto non è possibile inviare ness’ualtro comando.
Done	OUT:Bool	Esecuzione ultimo comando avvenuta con successo. Sia sul primo ciclo di scansione del PLC e per ogni errore di comunicazione anche se non si sta eseguendo un comando questo bit viene impostato a FALSE. Diversamente rimane sempre a TRUE a seguito di un’istruzione completata con successo fino alla successiva richiesta.
Error	OUT:Bool	Questo bit viene messo a TRUE per ogni errore di comunicazione e resettato ogni volta che si ripristina la comunicazione.
Return	OUT:String	Stringa di ritorno dal FlexyBowl. Nel caso di esecuzione di una QX verrà restituito il valore ‘Mooving’ fino al completamento. Una volta completata l’istruzione QX si troverà il valore dell’ultima richiesta di stato al FlexiBowl fino all’esecuzione successiva. In caso di errore di comunicazione verrà restituito il valore ‘Communication Error. Check state on Nettwork&Device ’
Return_ FlexiAlarm	OUT:Word	Word contentente l’eventuale codice di errore letto dal driver del FlexiBowl. Vedere la tabella allarmi

COMMAND LIST:

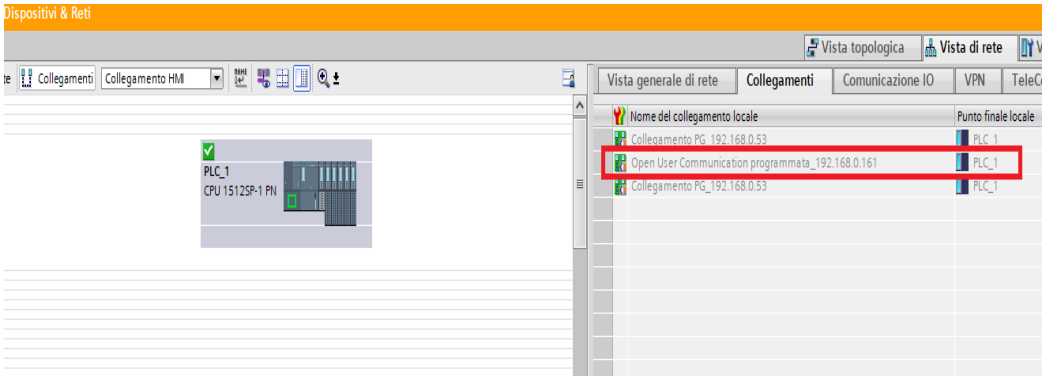
Comandi	Descrizione
QX2	Move
QX3	Move-Flip
QX4	Move-Flip-Blow
QX5	Move-Blow
QX6	Shake
QX7	Light on
QX8	Light off
QX9	Flip
QX10	Blow
QX11	Quick Emptying Option
QX12	Reset Alarm

Tabella Allarmi:

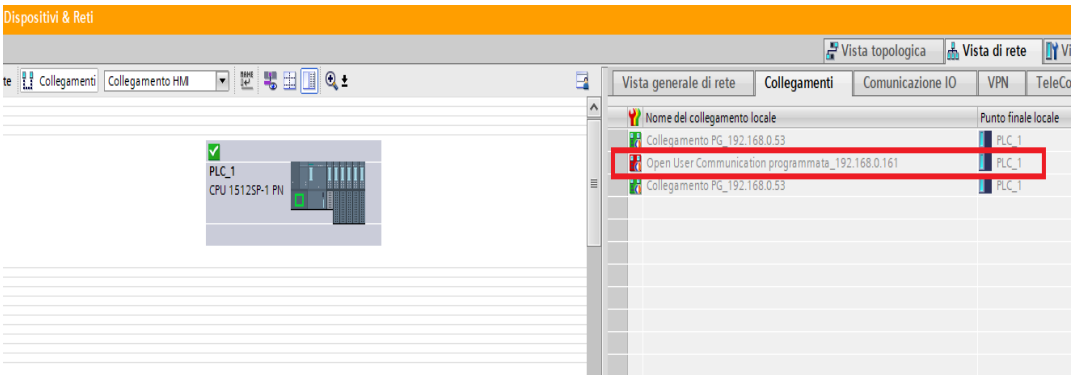
Hex Value	Descrizione
16#0001	Position Limit
16#0002	CCW Limit
16#0004	CW Limit
16#0008	Over Temp
16#0010	Excess Regen
16#0020	Over Voltage
16#0040	Under Voltage
16#0080	Over Current
16#0100	Open Motor Winding
16#0200	Bad Encoder
16#0400	Comm Error
16#0800	Bad Flash
16#1000	No Move
16#2000	Motor Resistance Out of Range
16#4000	Blank Q Segment
16#8000	(not used)

DIAGNOSTIC

Quando il blocco è riuscito ad aprire la comunicazione verso il FlexiBowl è possibile verificare online la presenza dell'icona della connessione stabilita nella sezione collegamenti all'interno della pagina della configurazione delle reti:



Diversamente quando c'è un errore di comunicazione con il dispositivo la situazione è la seguente:



TIA Portal V15

SCRIPT:

```
// Set the base parameter
// Convert the IP address into array of byte
#l := 1;
FOR #i := 1 TO 15 DO
   IF (#IPAddr[#i] <> '.') AND (#IPAddr[#i] <> '$00') THEN
      #string_app1 := CONCAT(IN1 := #string_app1, IN2 := #IPAddr[#i]);
   END_IF;
   IF (#IPAddr[#i] = '.') OR (#IPAddr[#i] = '$00') OR (#i = 15) THEN
      #Skt_Param_Snd.RemoteAddress.ADDR[#l] := UINT_TO_BYTE(IN := STRING_TO_UINT(#string_app1));
      #string_app1 := '';
      #l := #l + 1;
   END_IF;
   IF (#IPAddr[#i] = '$00') THEN
      EXIT;
   END_IF;
END_FOR;

// Set the IP address into array of byte
#Skt_Param_Snd.ConnectionType := 11;                  // 11=TCP
#Skt_Param_Snd.ActiveEstablished := TRUE;            // Stabilite active connection from the PLC
#Skt_Param_Snd.RemotePort := 7776;                   // FlexiBowl TCP listening port
#Skt_Param_Snd.ID := #Conn;                          // ID for connction parameter
#Skt_Param_Snd.LocalPort := 0;                       // Local port 0=Dinamic

#KeepAlive_Timer(IN := ((#SktTCPSend_instance.s_TDIAG_Status.State = 16#04) AND (NOT #Execute) AND (NOT
#InsideBusy) AND (NOT #ReadContSts)),
                 PT := T#20S);

IF #KeepAlive_Timer.Q THEN
   #ReadContSts := TRUE;
   #KeepAlive_Flag := TRUE;
END_IF;

// ###############################################################################################
#Exe_trig(CLK := #Execute);                          // Execution trigger

// Inizialize data
IF #Exe_trig.Q  AND (#SktTCPSendinstance.sTDIAGStatus.State = 16#04) THEN
   #InsideBusy := TRUE;
   #Done := FALSE;
   #ReadContSts := FALSE;
   #ReadAlarm := FALSE;
   #Error_Com := FALSE;
   #KeepAlive_Flag := FALSE;

   FOR #i := 0 TO 20 DO
      #SendSocketDat[#i] := 16#00;
   END_FOR;
   #SendSocketDat[1] := 16#07;

   FOR #i := 0 TO 51 DO
      #RcvSocketDat[#i] := 16#00;
   END_FOR;

   // ---------------------------------------------
   // Arrange the message to send
   // Headre   16#00 16#07
   // Message  ascii to hex message
   // Footer   16#0D

   FOR #i := 0 TO 19 DO
      #SendSocketDat[#i + 2] := #CMD[#i + 1];
      IF #SendSocketDat[#i + 2] = 16#00 THEN
         #SendSocketDat[#i + 2] := 16#0D;
         EXIT;
      END_IF;
   END_FOR;
// ---------------------------------------------
END_IF;
```

TIA Portal V15

SCRIPT:

```
#ReadAlm_trig(CLK := (NOT #ReadContSts));                                    // Execution read alarme trigger

// If ReadContSts funtion active send 'SC' command
IF #ReadContSts THEN
   #SendSocketDat[0] := 16#00;
   #SendSocketDat[1] := 16#07;
   #SendSocketDat[2] := 'S';
   #SendSocketDat[3] := 'C';
   #SendSocketDat[4] := 16#0D;
   FOR #i := 5 TO 19 DO
      #SendSocketDat[#i] := 16#00;
   END_FOR;
END_IF;

// If ReadAlarm funtion active send 'AL' command
IF #ReadAlm_trig.Q THEN
   #SendSocketDat[0] := 16#00;
   #SendSocketDat[1] := 16#07;
   #SendSocketDat[2] := 'A';
   #SendSocketDat[3] := 'L';
   #SendSocketDat[4] := 16#0D;
   FOR #i := 5 TO 19 DO
      #SendSocketDat[#i] := 16#00;
   END_FOR;
   #ReadAlarm := TRUE;
END_IF;

// ###############################################################################################
// Send data Section
// Calculate Message Size
FOR #i := 2 TO 19 DO
   IF (#SendSocketDat[#i] = 16#0D) THEN
      #SndSize:=INT_TO_UDINT(#i);
      EXIT;
   END_IF;
END_FOR;

#Snd_Trig(CLK := #SktTCPRcv_instance.DONE OR #KeepAlive_Timer.Q);            // Execution trigger
#SktTCPSend_instance(REQ := ((#Exe_trig.Q OR (#Snd_Trig.Q AND #ReadContSts) OR #ReadAlm_trig.Q)),
                     CONT := TRUE,
                     CONNECT := #Skt_Param_Snd,
                     DATA := #SendSocketDat,
                     LEN := #SndSize + 1);

// ###############################################################################################
// Receive data Section
#SktTCPRcv_instance(EN_R := TRUE,
                    CONT := TRUE,
                    CONNECT := #Skt_Param_Snd,
                    DATA := #RcvSocketDat,
                    LEN := 0,
                    ADHOC := TRUE);

// Decoding the message from the FlaxiBowl
IF #SktTCPRcv_instance.DONE AND (NOT #ReadAlarm) THEN
   #Return_Value := '';
   FOR #i := 2 TO 52 DO
      IF (#RcvSocketDat[#i] = 16#0D) THEN
         EXIT;
      END_IF;
      #Return_Value[#i - 1] := BYTE_TO_CHAR(#RcvSocketDat[#i]);
   END_FOR;
```

TIA Portal V15

SCRIPT:

```
    // If received data from FlaxiBowl (SC= 4 xxx) and ReadContSts funtion active, write 'QX Running' on Return_Value
variable elseif received data from FlaxiBowl (SC = xx 1x) and ReadContSts funtion active, write 'Mooving'
    //                                                  23   4 5678
    IF (#RcvSocketDat[7] = 16#31) AND #ReadContSts THEN
        #Return_Value := 'Mooving';
    END_IF;
    IF (#RcvSocketDat[5] = 16#34) AND #ReadContSts THEN
        #Return_Value := 'QX Running';
    END_IF;
END_IF;

// If the check FlexiBowl alarm it's done then set Done and reset Busy and terminate the function
IF #SktTCPRcv_instance.DONE AND #ReadAlarm THEN
    #h := 1;
    FOR #i := 5 TO 8 DO
        #string_app2[#h] := #RcvSocketDat[#i];
        #h := #h + 1;
    END_FOR;
    #Return_FlexiAlarm := STRING_TO_UINT(IN := #string_app2);
    #ReadAlarm := FALSE;
    #InsideBusy := FALSE;
    #Done := TRUE;
END_IF;

// If 'QX' command sent active ReadContSts funtion
IF #SktTCPRcv_instance.DONE AND (#SktTCPRcv_instance.RCVD_LEN <> 0) AND (#RcvSocketDat[2] = 16#25) AND
(NOT #ReadContSts) THEN
    #ReadContSts := TRUE;
END_IF;

// If received data from FlaxiBowl and not ReadContSts funtion active, set Done and reset Busy
IF #SktTCPRcv_instance.DONE AND (#SktTCPRcv_instance.RCVD_LEN <> 0) AND (NOT #ReadContSts) AND (NOT
#ReadAlarm) THEN
    #InsideBusy := FALSE;
    #Done := TRUE;
END_IF;

// If received data from FlaxiBowl (SC= 0 001) and ReadContSts funtion active, set Done and reset Busy and terminate the
function                                    234 5 678
IF #SktTCPRcv_instance.DONE AND (#RcvSocketDat[7] = 16#30) AND #ReadContSts THEN
    #ReadContSts := FALSE;
END_IF;

IF #KeepAlive_Flag AND (#SktTCPRcv_instance.RCVD_LEN <> 0) THEN
    #KeepAlive_Flag := FALSE;
    #Error_Com := FALSE;
END_IF;

// ##############################################################################################################
// TimeOut and error management
#Timeout_Timer(IN := (((NOT #ReadContSts) AND #InsideBusy) OR (#ReadContSts AND (NOT
#SktTCPRcv_instance.DONE)) OR #KeepAlive_Flag OR #ReadAlarm),
              PT := T#2S);

// Generate error
//
IF #Timeout_Timer.Q THEN
    #ReadContSts := FALSE;
    #ReadAlarm := FALSE;
    #KeepAlive_Flag := FALSE;
    #InsideBusy := FALSE;
    #Done := FALSE;
    #Error_Com := TRUE;
    #Return_Value := 'Communication Error. Check the flaxiBowl Con';
END_IF;

#Busy := #InsideBusy;
```