OMRON FLEXIBOWL PLUGIN

Questo Plugin è nato con l'idea di comunicare in maniera rapida e sicura con il FlexiBowl® tramite i PLC Omron Serie N con ethernet a bordo, mediante l' utilizzo di istruzioni in linuaggio LD/ST.
Il Plugin sviluppato Sysmac Studio 1.25 non necessita di una licenza aggiuntiva ed è sviluppato per comunicare attraverso la scheda ethernet integrata dei PLC Omron.

Richiede versione firmware 1.16

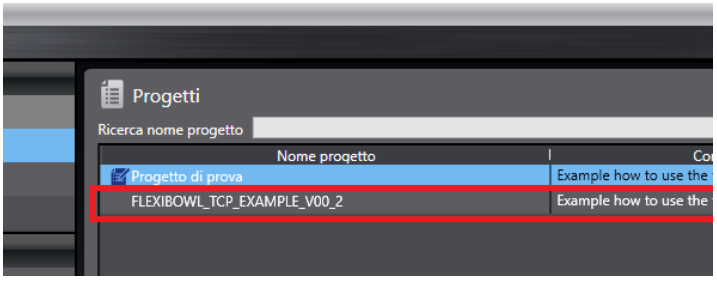
FlexiBowl®

Sysmac Studio

STEP 1:

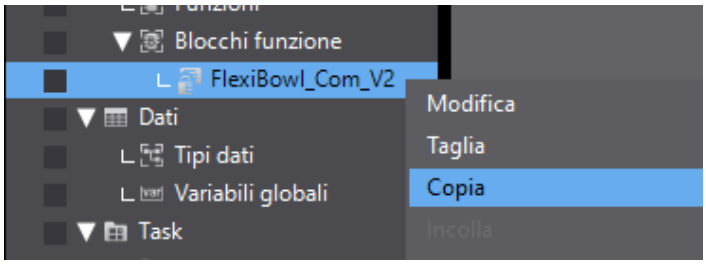
Scompattare il progetto di esempio cliccando due volte sul file “FLEXIBOWL_TCP_EXAMPLE_V00_3”

STEP 2:



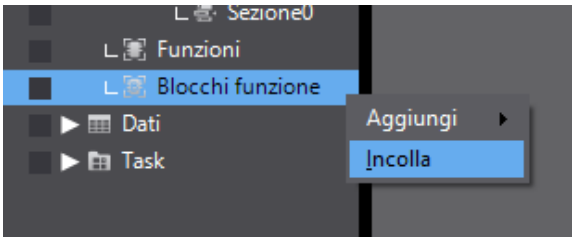
Una volta scompattato aprire il progetto dalla sezione “Apri progetto” di Sysmac Studio

STEP 3:



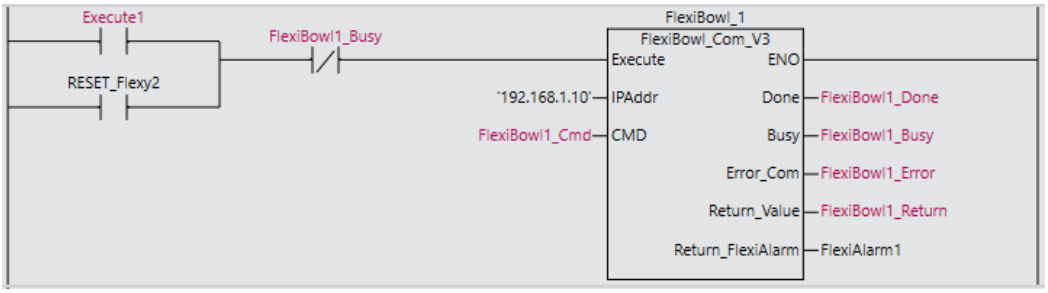
Copiare il blocco funzione “FlexiBowl_Com_V3”

STEP 4:



Cliccare nel proprio progetto con il tasto destro del mouse nella sezione “Blocchi funzione” all’interno dell’albero di progetto e cliccare “Incolla”

STEP 5:



Inserire ora nel proprio software il blocco funzione e popolare gli ingressi e le uscite del blocco

TAG	Tipo	Descrizione
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. Es: ‘192.168.0.161’ 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.
CMD	IN:String	Comando in formato stringa da inviare al flexiBowl (vedere tabella più avanti)

STEP 5:

TAG	Tipo	Descrizione
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 the flaxiBowl Con'
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)

Sysmac Studio

SCRIPT:

```
// If soket colsed then try to open soket
Time_Open(In:= (NOT Socket_Opened) AND (NOT Time_Open.Q),PT:=T#1S);
Open_trig(Clk:= Time_Open.Q);

SktTCPConnect_instance(                                  // Request a connection.
                        Execute:=Open_trig.Q,
                        SrcTcpPort:=UINT#0,              // Local TCP port number: Automatically assigned.
                        DstAdr:= IPAddr,                 // Remote IP address
                        DstTcpPort:=UINT#7776,           // Destination TCP port number
                        Socket=>WkSocket);               // Socket

 IF SktTCPConnect_instance.Done THEN                     // If the socket is open
                        Socket_Opened:=TRUE;
                        Error_Com := FALSE;
END_IF;

 KeepAlive_Timer(In:= (Socket_Opened AND (NOT Execute) AND (NOT Busy) AND (NOT
KeepAlive_Timer.Q)),PT:=T#20S);
 IF KeepAlive_Timer.Q THEN
            ReadContSts:=TRUE;
            KeepAlive_Flag:=TRUE;
 END_IF;

// ##########################################################################################################
// Inizialize data for normal
Exe_trig(CLK := Execute);      // Execution trigger

IF Exe_trig.Q  AND Socket_Opened THEN
   Busy := 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;
```

Sysmac Studio

SCRIPT:

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

            ToAryByte(In:=CMD, Order:=_LOW_HIGH, AryOut:=SendSocketDat[2]);          // Convert text command
into array byte
   FOR i := 2 TO 19 DO
      IF SendSocketDat[i] = 16#00 THEN
         SendSocketDat[i] := 16#0D;
         EXIT;
      END_IF;
   END_FOR;
END_IF;
// ---------------------------------------------

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] := 16#53;
   SendSocketDat[3] := 16#43;
   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] := 16#41;
   SendSocketDat[3] := 16#4C;
   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 SndSize := 2 TO 19 DO
            IF (SendSocketDat[SndSize] = 16#0D) THEN EXIT;
   END_IF;
END_FOR;

// Send data
Snd_Trig(CLK := SktTCPRcv_instance.Done OR KeepAlive_Timer.Q);        // Execution trigger
SktClearBuff(Execute:=((Exe_trig.Q OR (Snd_Trig.Q AND ReadContSts) OR ReadAlm_trig.Q) AND
Socket_Opened),
            Socket:=WkSocket);

SktTCPSend_instance(
            Execute:=SktClearBuff.Done,
            Socket :=WkSocket,                        // Socket
            SendDat:=SendSocketDat[0],                // Send data
            Size:= SndSize + 1);                      // Send data size
```

Sysmac Studio

SCRIPT:

```
// ###############################################################################################################
// Receive data Section
Rcv_Trig(CLK:= SktTCPSend_instance.Done);      // Execution trigger

SktTCPRcv_instance(
        Execute:=Rcv_Trig.Q,
        Socket :=WkSocket,                    // Socket
        TimeOut:=UINT#0,                      // Timeout time
        Size   :=UINT#52,                     // Receive data size
        RcvDat :=RcvSocketDat[0]);            // Receive data

// 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;
   END_FOR;
            Return_Value:=AryToString(In:=RcvSocketDat[2], Size:=i-2);

// 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'

            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;
```

Sysmac Studio

SCRIPT:

```
// 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
		Return_FlexiAlarm:=STRING_TO_WORD(MID(AryToString(In:=RcvSocketDat[5], Size:=4),UINT#4, UINT#1));
		ReadAlarm := FALSE;
		Busy := FALSE;
   Done := TRUE;
END_IF;

// If 'QX' command sent active ReadContSts funtion
IF SktTCPRcv_instance.DONE AND (SktTCPRcv_instance.RcvSize <> 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.RcvSize <> 0) AND (NOT ReadContSts) AND (NOT ReadAlarm)
THEN
   Busy := 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
IF SktTCPRcv_instance.Done AND (RcvSocketDat[7] = 16#30) AND ReadContSts THEN
   ReadContSts := FALSE;
END_IF;

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

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

// Generate error

SktClose_instance(
		Execute:=Timeout_Timer.Q,
		Socket:=WkSocket);

IF Timeout_Timer.Q THEN
		Socket_Opened:=FALSE;
   ReadContSts := FALSE;
		ReadAlarm := FALSE;
		KeepAlive_Flag:=FALSE;
   Busy := FALSE;
   Done := FALSE;
   Error_Com := TRUE;
   Return_Value := 'Communication Error. Check the flaxiBowl Con';
END_IF;
```