SIEMENS FLEXIBOWL PLUGIN

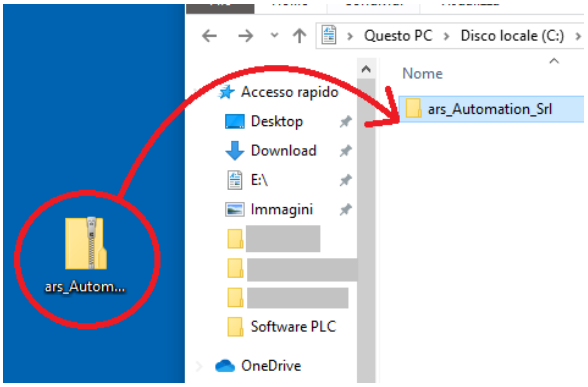
This plugin was created with the idea of communicating quickly and safely with the FlexiBowl® throuth the Siemens PLCs S7-1200 and S7-1500 series using KOP/SCL instructions language.
Whit the plugin developed in TIA Portal V15 doesn't required additional licenses and it was develop for comunicate throuth the ethernet board integrated in PLCs Siemens.

Requited firmware 4.0 version or higher for PLC S7-1200
Requited firmware 2.0 version or higher for PLC S7-1500
Requited firmware 2.0 version or higher for PLC ET200SP

FlexiBowl®

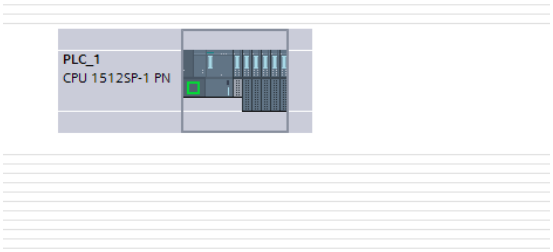
TIA Portal V15

STEP 1:

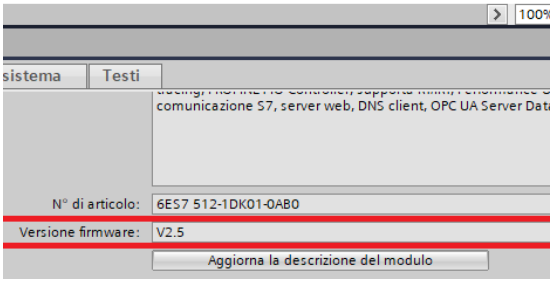


Unpackage the "ars_Automation_Srl" Library in a desired folder

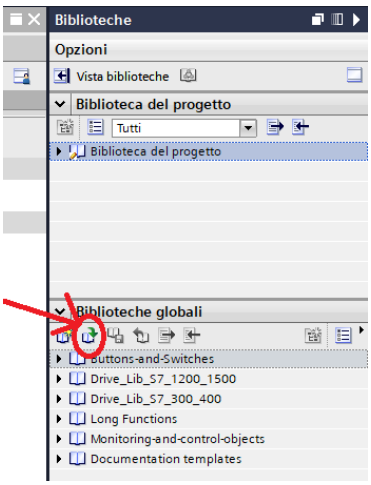
STEP 2:



Make sure in the hardware configuration of the TIA Portal that in the CPU firmware version it is greater than or equal to 2.0 for the S7-1500 or ET200 SP PLCs or greater than or equal to 4.0 for the S7-1200 PLCs

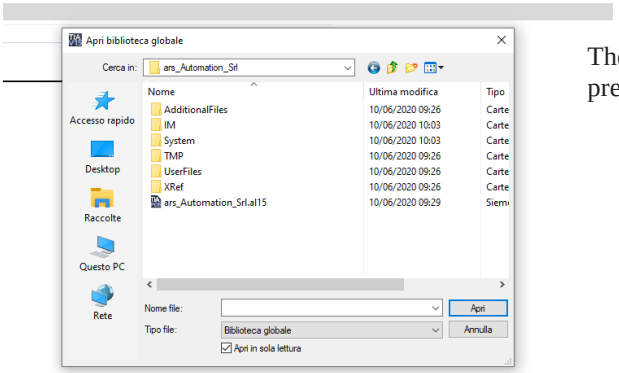


STEP 3:



Click on the "Open global library" icon inside the section library in the TIA portal.

STEP 4:



Then select the library file within the previously unzipped folder.

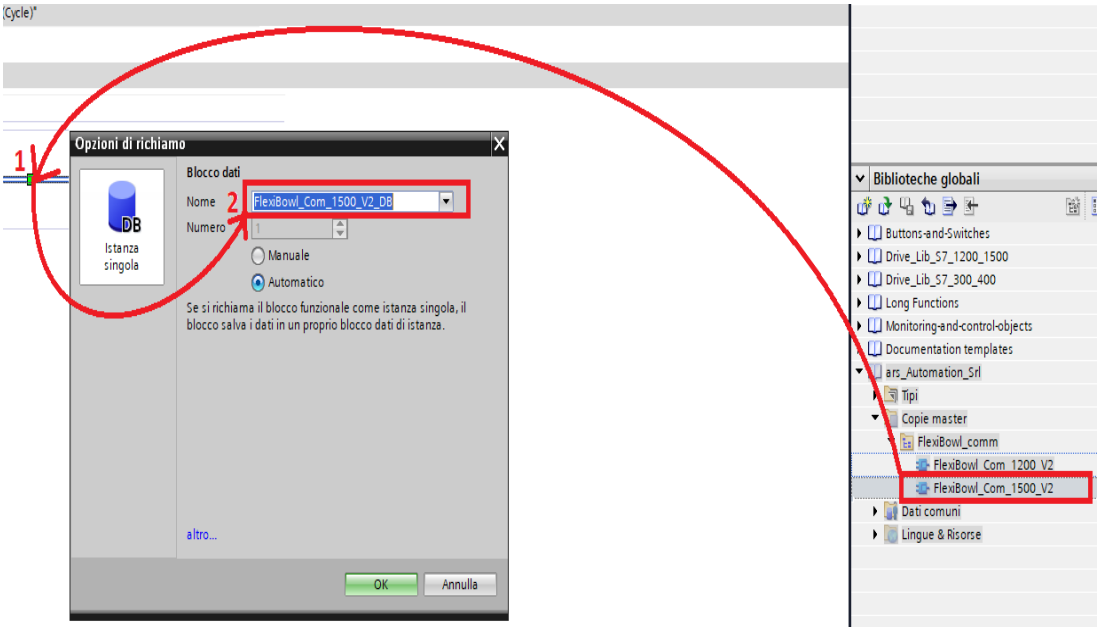
STEP 5:

Drag the communication block now available in the library into the software; select the block according to the CPU you are using:

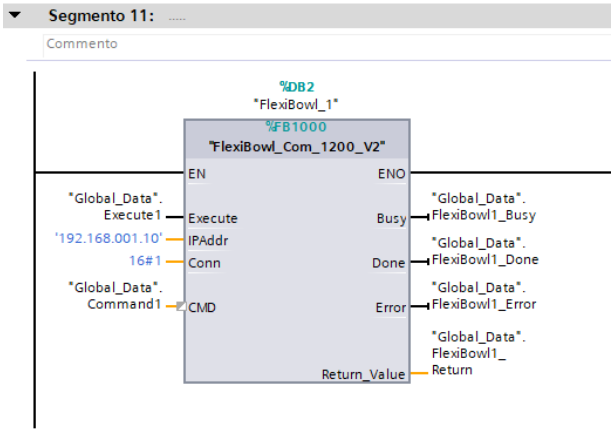
- FlexiBowl_com_1200_V2 for S7-1200;
- FlexiBowl_com_1500_V2 for S7-1500 or ET200 SP;

Once the block has been inserted into the software, you will be asked to give an instance DB. Multi-instance blocks can also be created.

NB: It is recommended to use only one block for each FlexiBowl you intend to communicate and manage the requests to be sent upstream of the block, using the "Done" and "Busy" outputs of the block



STEP 6:



Now populate the block inputs and outputs according to your needs:

TAG	Type	Description
EN	IN:Bool	Block Enable. It must always be enabled because inside the block there are instructions that keeping the TCP connection active
Execute	IN:Bool	Send the present command on the "CMD" input. The block considerate this input into only for one scan of PLC cycle and if communication is established.
IPAddr	IN:String	IP address of the FlexiBowl. Specify the IP address of the Flexibowl. The length of the string is constrained, therefore it is necessary to fill all 4 triads with three numbers eg: 192.168.0.161 → '192.168.000.161' 10.100.120.5 → '010.100.120.005' The TCP communication instructions are high level and managed by the PLC ethernet soket therefore even if the FlexiBowl is in another subnet and there is a reference gateway set in the hardware configuration of the PLC it is possible to directly specify the final FlexiBowl IP address.

STEP 6:

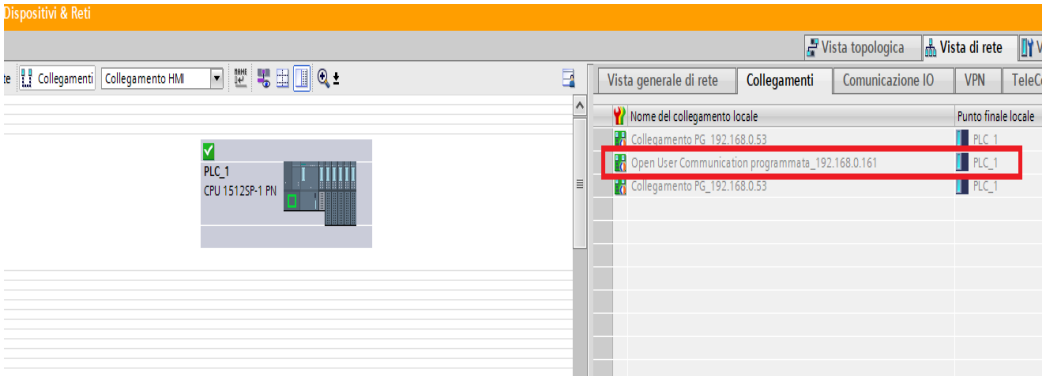
TAG	Type	Description
CMD	IN:String	Command in string format to be sent to the flexiBowl (see table below)
Conn	IN: CONN_OUC	Connection ID, this parameter must be hexadecimal ex: “16#01” and must be unique and including between 1 and the maximum number of TCP connection supported by the PLC ex 32
ENO	OUT:Bool	Block execution successful without any software error
Busy	OUT:Bool	QX execution in progress by FlexiBowl, therefore no other command can be sent.
Done	OUT:Bool	Last command execution successful. Both on the first scan cycle of the PLC and for each communication error even if a command is not being executed, this bit is set to FALSE. Otherwise it always remains at TRUE following a successfully completed instruction until the next request.
Error	OUT:Bool	This bit is set to TRUE for each communication error and reset each time communication is restored.
Return	OUT:String	Return string from FlexyBowl. In the case of execution of a QX the '**Mooving**' value will be returned until completion. Once the QX instruction has been completed, the value of the last status request to the FlexiBowl will be found until the next execution. In case of a communication error, the value ‘**Communication Error. Check state on Nettwork&Device**’ will be returned.

COMMAND LIST:

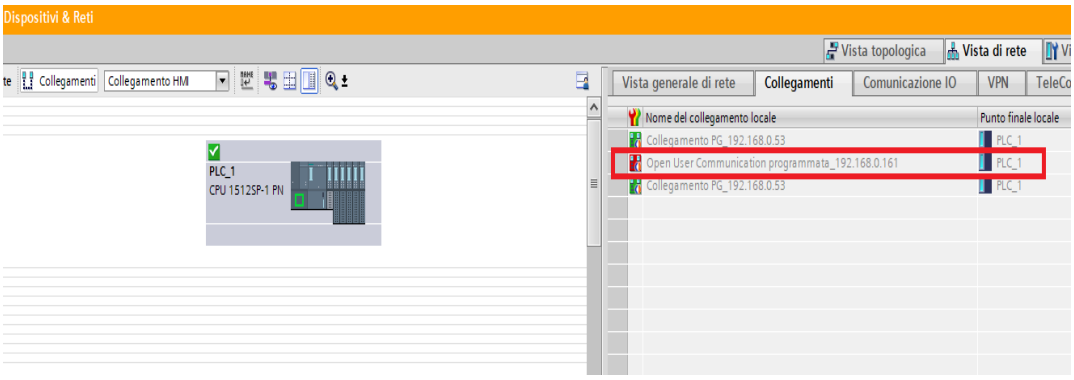
Command	Description
QX2	Move
QX3	Move-Flip
QX4	Move-Flip-Blow
QX5	Move-Blow
QX6	Shake
QX7	Light on
QX8	Light off
QX9	Blow
QX10	Flip
QX11	Quick Emptying Option
QX12	Reset Alarm

DIAGNOSTIC

When the block has managed to open the communication to the FlexiBowl, it is possible to check online for the presence of the connection icon established in the connections section within the network configuration page of TIA Portal:



Otherwise, when there is a communication error with the device, the situation is like follows:



TIA Portal V15

SCRIPT:

```
// Set the base parameter
// Convert the IP address into array of byte
#Skt_Param_Snd.RemoteAddress.ADDR[1] := UINT_TO_BYTE(IN := STRING_TO_UINT(CONCAT(IN1 := CONCAT(IN1
:= #IPAddr[1], IN2 := #IPAddr[2]), IN2 := #IPAddr[3])));
#Skt_Param_Snd.RemoteAddress.ADDR[2] := UINT_TO_BYTE(IN := STRING_TO_UINT(CONCAT(IN1 := CONCAT(IN1
:= #IPAddr[5], IN2 := #IPAddr[6]), IN2 := #IPAddr[7])));
#Skt_Param_Snd.RemoteAddress.ADDR[3] := UINT_TO_BYTE(IN := STRING_TO_UINT(CONCAT(IN1 := CONCAT(IN1
:= #IPAddr[9], IN2 := #IPAddr[10]), IN2 := #IPAddr[11])));
#Skt_Param_Snd.RemoteAddress.ADDR[4] := UINT_TO_BYTE(IN := STRING_TO_UINT(CONCAT(IN1 := CONCAT(IN1
:= #IPAddr[13], IN2 := #IPAddr[14]), IN2 := #IPAddr[15])));

// 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

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

// Inizialize data
IF #Exe_trig.Q  AND (#SktTCPSend_instance.s_TDIAG_Status.State = 16#04) THEN
   #Busy := TRUE;
   #Done := FALSE;
   #ReadContSts := FALSE;
   #Error := 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:

```
// 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;
END_IF;

//#############################################################################################
// Send data Section
#SktTCPSend_instance(REQ := #Exe_trig.Q OR (#SktTCPRcv_instance.DONE AND #ReadContSts),
                     CONT := TRUE,
                     CONNECT := #Skt_Param_Snd,
                     DATA := #SendSocketDat,
                     LEN := 20);

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

// Encoding the message from the FlaxiBowl
IF #SktTCPRcv_instance.DONE THEN
   #Return_Value:='';
   FOR #i := 2 TO 52 DO
      IF (#Return_Value[#i] = BYTE_TO_CHAR(16#0D)) THEN
         EXIT;
      END_IF;
      #Return_Value[#i - 1] := BYTE_TO_CHAR(#RcvSocketDat[#i]);
   END_FOR;
END_IF;

// If 'QX' command sent active ReadContSts funtion
IF (#SendSocketDat[2] = 'Q') AND (#SendSocketDat[3] = 'X') AND (#RcvSocketDat[2] = '%') 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) THEN
   #Busy := FALSE;
   #Done := TRUE;
END_IF;

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

TIA Portal V15

SCRIPT:

```
// If received data from FlaxiBowl (SC= 4 019) and ReadContSts funtion active, write 'Mooving' on Return_Value variable
//                                          234 5 678
IF (#RcvSocketDat[5] = 16#34) AND #ReadContSts THEN
   #Return_Value := 'Mooving';
END_IF;

//
########################################################################################################
###############
// TimeOut and error management
// NB:  Occorre filtrare con un timer di almeno 10S l'errore di connessione, poichè il driver del FlexiBowl
//      dopo circa 1 minuto che è fermo senza nessun comando, chiude la connessione, a questo punto il PLC
//      va in errore ma ripristina immediatamente la connessione poichè se ne occupa il socket, quindi ricevo
//      un errore di comunicazione per qualche secondo.
#Timeout_Timer(IN := ((NOT #ReadContSts) AND #Busy) OR (#SktTCPSend_instance.s_TDIAG_Status.State <> 16#04),
               PT := T#10S);

// Reset for OK connection
#Rst_trig(CLK := (NOT #Timeout_Timer.Q));
IF #Rst_trig.Q THEN
   #Error := FALSE;
   #Return_Value := '';
END_IF;

// Generate error
IF #Timeout_Timer.Q THEN
   #ReadContSts := FALSE;
   #Busy := FALSE;
   #Done := FALSE;
   #Error := TRUE;
   #Return_Value := 'Communication Error. Check state on Nettwork&Device';
END_IF;
```