Manual

FlexiBowl®

SCHNEIDER MODICON M241, M251 AND M262 PLUG-IN

ars | Feeding Industrial Robotics

INDEX

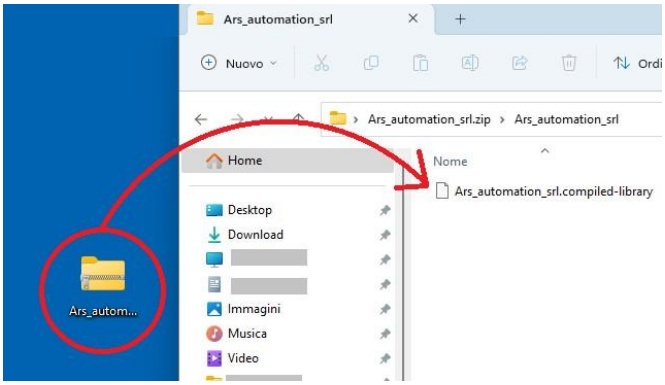
Questo Plugin è nato con l'idea di comunicare in maniera rapida e sicura con il FlexiBowl® tramite i **PLC Schneider Serie Modicon M241, M251 ed M262**, mediante l'utilizzo di istruzioni in **linguaggio LD/ST**.

Il Plugin sviluppato in **Machine Expert V2.1** non necessita di una licenza aggiuntiva.

FlexiBowl® Plug-In

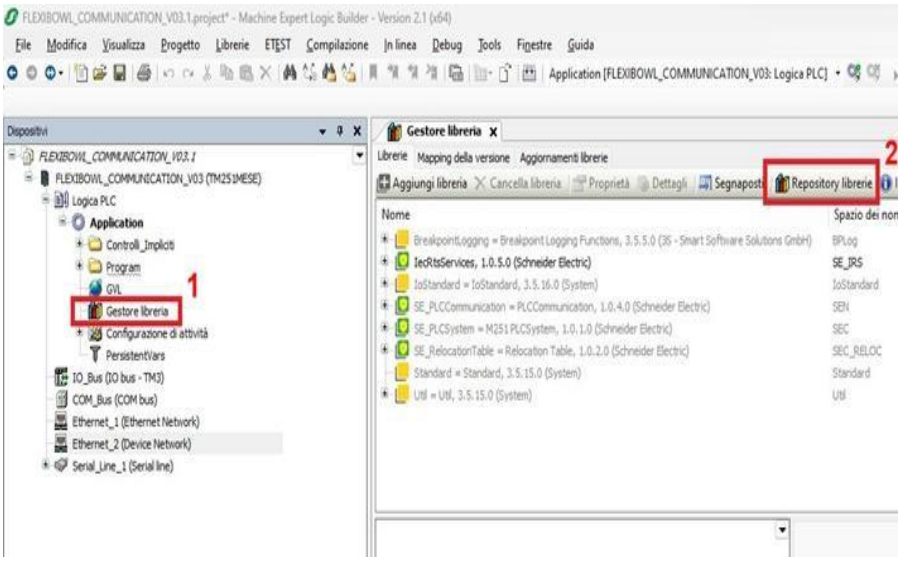
FlexiBowl by ars

STEP 1:



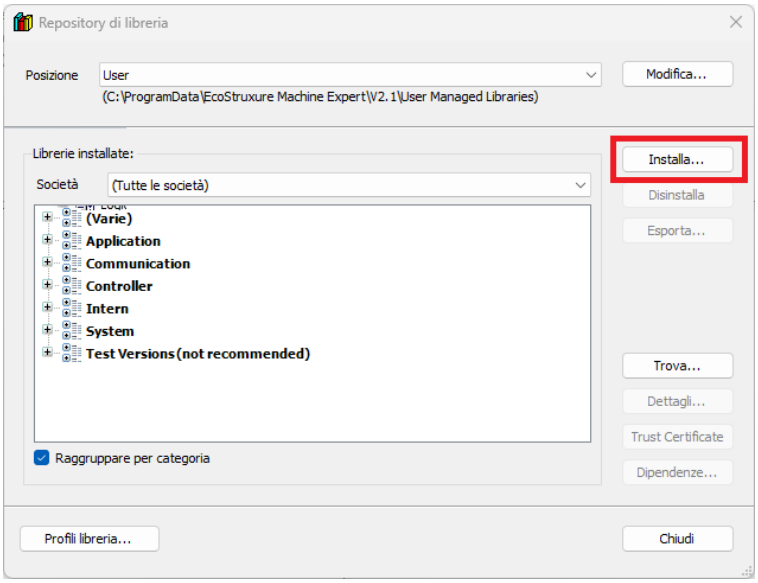
Scompattare la Libreria "ars_Automation_Srl" in una cartella desiderata

STEP 2:



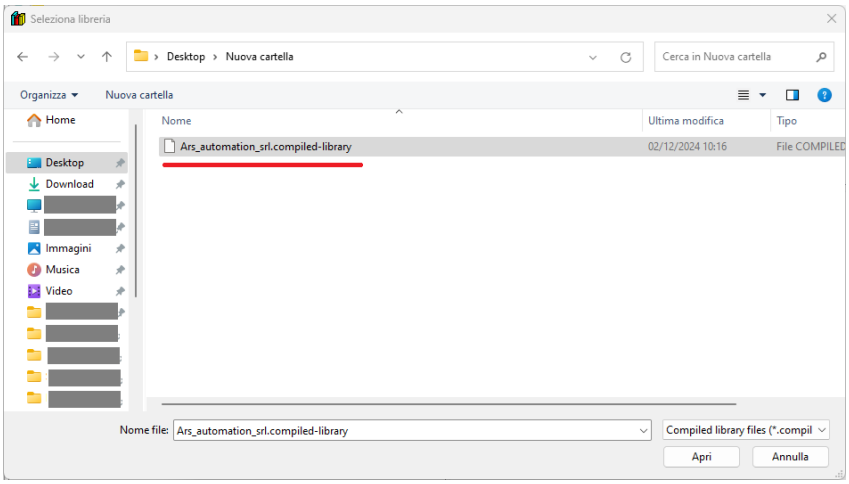
Aprire ora dal proprio progetto il "Repository librerie"

STEP 3:



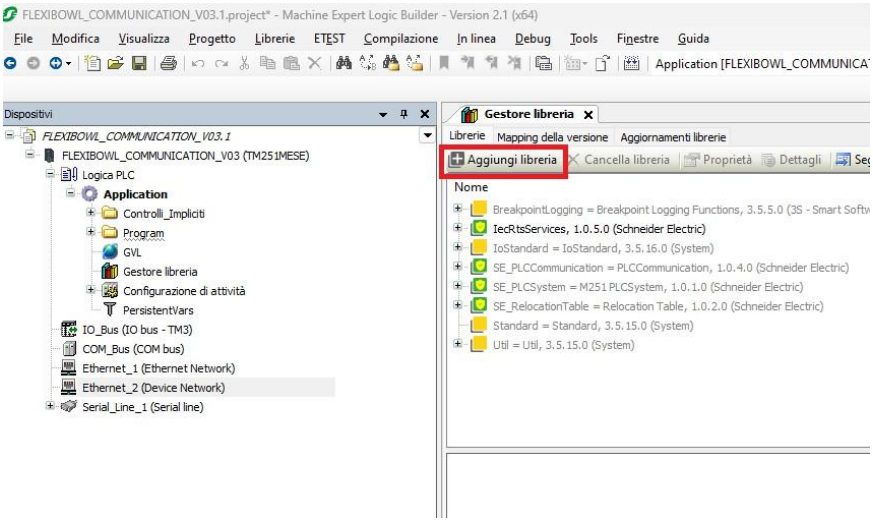
Cliccare sul pulsante installa

STEP 4:



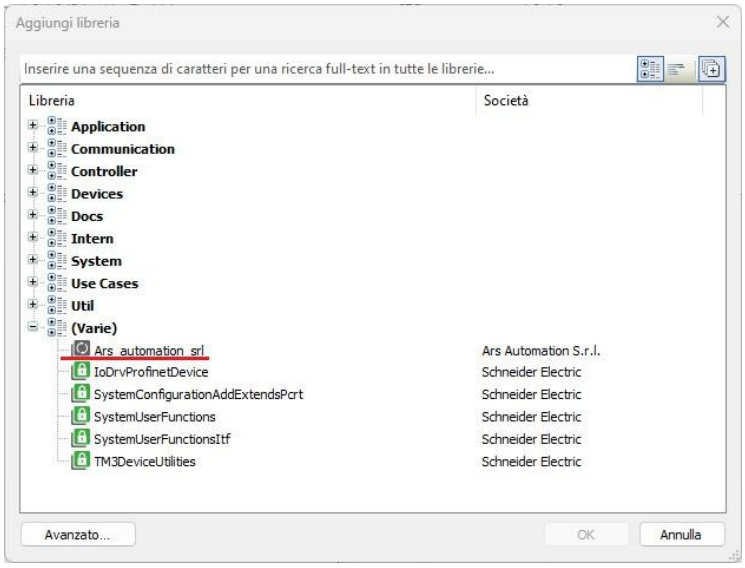
Selezionare quindi il file della libreria precedentemente scompattata e cliccare su "Apri". In seguito chiudere il Repository librerie.

STEP 5:



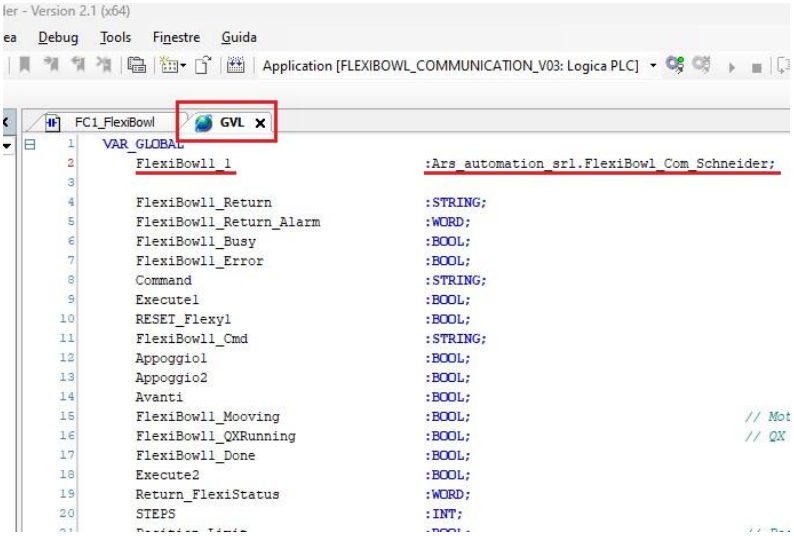
Sempre all'interno del Vostro progetto cliccare ora su "Aggiungi libreria"

STEP 6:



Espandere l'albero delle librerie alla voce "(Varie)", selezionare quindi "Ars_automation_srl" e cliccare su OK

STEP 7:



Nelle "GVL" o Variabili Globali dichiarare la propria istanza facente riferimento al seguente tipo: "**Ars_automation_srl.FlexiBowl_Com_Schneider**";

FlexiBowl by ars

STEP 8:

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. 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.

FlexiBowl by ars

STEP 8:

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

FlexiBowl by ars

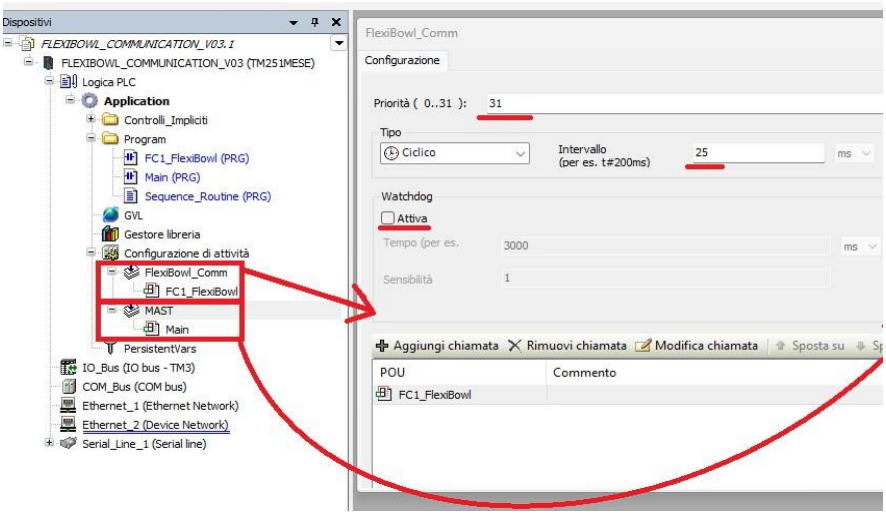
COMMAND LIST:

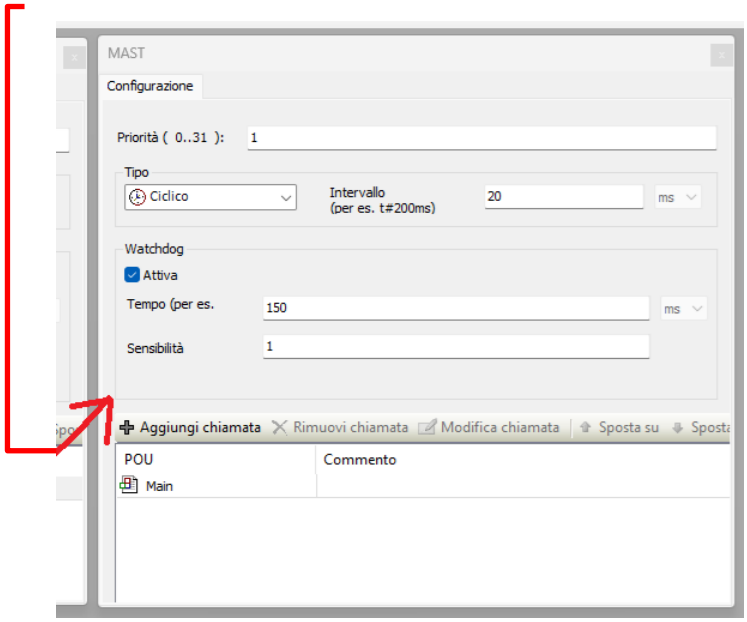
Comandi	Descrizione
QX2	Move
QX3	Move-Flip
QX4	Move-Flip-Blow
QX5	Move-Blow
QX6	Shake
QX7	Light on
QX8	Light off
QX9	Flip
QX10	Blow
QX11	Quick Emptying Option
QX12	Reset Alarm

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)

RACCOMAN DAZIONI



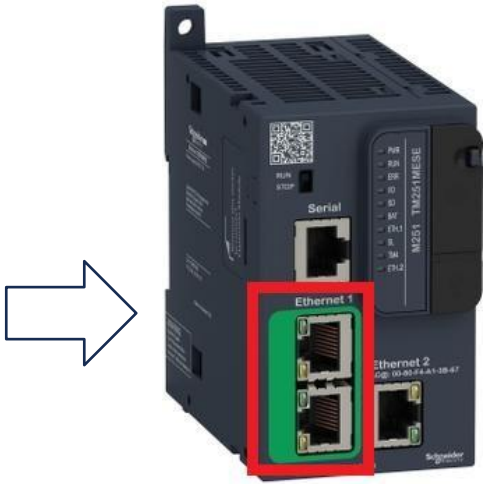


Il blocchetto funzionale "**FlewiBowl_Com_Schneider**" trattato nel presente manuale sfrutta alcune librerie di sistema Codesys che possono avere effetti bloccanti sull'esecuzione dei task del PLC.
Pertanto **è fortemente consigliato** creare un task totalmente dedicato alla routine che si occupa della comunicazione con il FlexiBowl, a tempo di esecuzione "**ciclico**" e minimo intervallo a "**25 ms**" con Watchdog "**disabilitato**" e la priorità più bassa possibile, nel caso sopra riportato per esempio è stata impostata a "**31**".

Questo evita arresti indesiderati del programma utente in caso di errori generati dalle istruzioni utilizzate all'interno del blocco "**FlewiBowl_Com_Schneider**"

RACCOMANDAZIONI

N.B. Nel caso si utilizzi un PLC **M251MESE** le librerie Codesys di comunicazione fanno riferimento alla porta "**Ethernet 1**" ovvero quella in modalità switch come evidenziato in foto. Pertanto il FlexiBowl deve essere interconnesso su tale porta.



FlexiBowl by ars

SCRIPT:

```
// ###############################################################################################
// Set the base parameter
// Convert the IP address
syssocket.SysSockInetAddr(IPAddr,  ADR(_ipAddr.sin_addr));

// Set the IP address into array of byte
_ipAddr.sin_port := syssocket.SysSockHtons(7776);                    // FlexiBowl TCP listening port
_ipAddr.sin_family := SOCKET_AF_INET;                                // Codesys Identification number of address family


// If the connection TCP are close, then try to connect with the flexybowl
KeepAlive_Timer(IN := (((SokStatus = FLEXYSOCK_CONNECTED) OR (SokStatus = FLEXYSOCK_IDLE)) AND (NOT
Execute) AND (NOT InsideBusy) AND (NOT ReadContSts) AND (NOT ReadAlarm)),
              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 ((SokStatus = FLEXYSOCK_CONNECTED) OR (SokStatus = FLEXYSOCK_IDLE)) 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];
      IF SendSocketDat[i + 2] = 16#00 THEN
         SendSocketDat[i + 2] := 16#0D;
         EXIT;
      END_IF;
   END_FOR;
//
END_IF
```

FlexiBowl by ars

SCRIPT:

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

// If ReadContSts funtion active send 'SC' command
IF ReadContSts THEN
        _Char := 'SC';
    SendSocketDat[0] := 16#00;
    SendSocketDat[1] := 16#07;
    SendSocketDat[2] := _Char[0];
    SendSocketDat[3] := _Char[1];
    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
        _Char := 'AL';
    SendSocketDat[0] := 16#00;
    SendSocketDat[1] := 16#07;
    SendSocketDat[2] := _Char[0];
    SendSocketDat[3] := _Char[1];
    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_Trig1(CLK := (SokStatus = FLEXYSOCK_IDLE));            // Execution trigger 1
Snd_Trig2(CLK := (KeepAlive_Timer.Q));                     // Execution trigger 2

CASE SokStatus OF
// ############################## CREATING ##############################
        FLEXYSOCK_CLOSE:
                SokStatus := FLEXYSOCK_CONNECTING;

                syssocket.SysSockClose(SocketId);
                SocketId := syssocket.SysSockCreate(SOCKET_AF_INET, SOCKET_STREAM,
                        SOCKET_IPPROTO_TCP , ADR(SockCreateResult));

                SysSockIoctl(SocketId, SOCKET_FIONBIO, ADR(enable_option));

                IF (SocketId = RTS_INVALID_HANDLE) THEN
                        SokStatus := FLEXYSOCK_ERROR;
                ELSE
                        SokStatus := FLEXYSOCK_CONNECTING;
                END_IF
```

FlexiBowl® by ars

SCRIPT:

```
// ############################## CONNECTING ##############################
        FLEXYSOCK_CONNECTING:
                SockConnectResult := SysSockConnect(SocketId, ADR(_ipAddr), SIZEOF(_ipAddr));

            IF (SockConnectResult <> RTS_INVALID_HANDLE) THEN
                SokStatus := FLEXYSOCK_CONNECTED;
            ELSE
                SokStatus := FLEXYSOCK_ERROR;
            END_IF

// ############################## SENDING ##############################
FLEXYSOCK_CONNECTED, FLEXYSOCK_IDLE:
SockReceiveDone := FALSE;
IF ((Exe_trig.Q OR ((Snd_Trig1.Q OR Snd_Trig2.Q) AND ReadContSts) OR ReadAlarm)) THEN
        SockSendDone := FALSE;
        SentBytes := 0;
        SokStatus := FLEXYSOCK_SENDBUSY;
        SentBytes := syssocket.SysSockSend(SocketId, ADR(SendSocketDat), SndSize + 1, SOCKET_MSG_NONE,
ADR(SockSendResult));

        IF ((SentBytes > 0) AND (SockSendResult <> RTS_INVALID_HANDLE)) THEN
            SokStatus := FLEXYSOCK_RECEIVEREQ;
            SockSendDone := TRUE;
        END_IF;
        IF (SockSendResult = RTS_INVALID_HANDLE) THEN
            SokStatus := FLEXYSOCK_ERROR;
            SockSendDone := FALSE;
        END_IF
END_IF;

// ############################## RECEIVING ##############################
// Receive data Section
FLEXYSOCK_RECEIVEREQ:
        RcvBytes := 0;
        SokStatus := FLEXYSOCK_RECEIVBUSY;
        FOR i := 0 TO 51 DO
    RcvSocketDat[i] := 16#00;
  END_FOR;

        RcvBytes := syssocket.SysSockRecv(SocketId, ADR(RcvSocketDat), SIZEOF(RcvSocketDat),
SOCKET_MSG_NONE, ADR(iecRcvResult));

        IF ((RcvBytes > 0) AND (RcvSocketDat[0] = 16#00) AND (RcvSocketDat[1] = 16#07) AND (iecRcvResult <>
RTS_INVALID_HANDLE)) THEN
            SokStatus := FLEXYSOCK_IDLE;
            SockReceiveDone := TRUE;
        END_IF;
        IF (iecRcvResult = RTS_INVALID_HANDLE) THEN
            SokStatus := FLEXYSOCK_ERROR;
            SockReceiveDone := FALSE;
        END_IF;
END_CASE;
// ##########################################################################################

// Decoding the message from the FlaxiBowl
IF SockReceiveDone AND (NOT ReadAlarm) THEN
        FOR i := 0 TO 51 DO
            Return_Value[i] := 16#00;
        END_FOR;
  FOR i := 2 TO 52 DO
    IF (RcvSocketDat[i] = 16#0D) THEN
      EXIT;
    END_IF;
            Return_Value[i - 2] := RcvSocketDat[i];
  END_FOR;
```

FlexiBowl by ars

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'
   //                                          234 5 678
   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 SockReceiveDone 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(string_app2);   ReadAlarm :=
   FALSE;
   InsideBusy := FALSE;
   Done :=
TRUE;   END_IF;

// If 'QX' command sent active ReadContSts funtion
IF SockReceiveDone AND (RcvBytes <> 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 SockReceiveDone AND (RcvBytes <> 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 SockReceiveDone AND (RcvSocketDat[7] = 16#30) AND
   ReadContSts THEN   ReadContSts := FALSE;
END_IF;

IF KeepAlive_Flag AND (RcvBytes <> 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
SockReceiveDone)) OR   KeepAlive_Flag OR ReadAlarm),
          PT := T#5S);

// Generate error
IF Timeout_Timer.Q OR (SokStatus = FLEXYSOCK_ERROR)
   THEN   ReadContSts := FALSE;
   ReadAlarm := FALSE;
   KeepAlive_Flag :=
   FALSE;   InsideBusy :=
   FALSE;   Done := FALSE;
   Error_Com := TRUE;
   Return_Value := 'Communication Error. Check the
            flaxiBowl Con';
            syssocket.SysSockClose(SocketId);
            SokStatus := FLEXYSOCK_CLOSE;
END_IF;

Busy := InsideBusy;
```