Manual

FlexiVision

KUKA PLUG-IN

ars | Feeding Industrial Robotics

INDICE

Questo Plugin è nato con l'idea di comunicare in maniera **rapida e sicura** con Flexivision tramite i robot **Kuka.**
Il Plugin, per il funzionamento, necessita della **licenza KUKA Ethernet KRL**.

FlexiBowl® Plug-In

Configurazione del Plug-In

Step 1.

Panoramica KUKA.Ethernet KRL

Funzioni KUKA.Ethernet KRL è un pacchetto tecnologico ricaricabile con le seguenti funzioni:

- Scambio di dati tramite l'**EKI**
- Ricezione di dati **XML** di un sistema esterno
- Invio di dati **XML** a un sistema esterno
- Ricezione di dati binari di un sistema esterno
- Invio di dati binari a un sistema esterno

Proprietà

- Controllo robot e sistema esterno come client o server
- Configurazione di collegamenti tramite il file di configurazione basato su XML
- Configurazione di **"Messaggi evento"**
- Controllo di collegamenti tramite un ping sul sistema esterno.
- Lettura e scrittura di dati dell'interprete Submit
- Lettura e scrittura di dati dell'interprete robot

Comunicazione I dati vengono trasferiti tramite il protocollo **TCP/IP**. L'utilizzo del protocollo **UDP/IP** è possibile, ma non consigliato (protocollo di rete privo di collegamento, ad es. nessun riconoscimento della perdita di dati).

Configurazione del Plug-In

Step 2.

Configurazione di un collegamento Ethernet

Panoramica

Un collegamento Ethernet viene configurato tramite un file **XML**. per ogni collegamento, nella cartella **C:\KRC\ROBOTER\Config\User\Common\EthernetKRL** del controllo robot deve essere definito un file di configurazione.
Il nome del file XML è al tempo stesso la chiave di accesso in **KRL**.
Esempio: ...\EXT.XML —> EKI_INIT("EXT")

Struttura XML per caratteristiche di collegamento

Descrizione

Nella sezione **<EXTERNAL> ... </EXTERNAL>**, possono essere definite le impostazioni per il sistema esterno:
I file XML sono **"case sensitive"** . Considerare le maiuscole/minuscole.

```
<ETHERNETKRL>
        <CONFIGURATION>
            <EXTERNAL></EXTERNAL>
            <INTERNAL></INTERNAL>
        </CONFIGURATION>
        <RECEIVE>
            <ELEMENTS></ELEMENTS>
        </RECEIVE>
        <SEND>
            <ELEMENTS></ELEMENTS>
        </SEND>
</ETHERNETKRL>
```

Configurazione Plug-In

Step 3.

Di seguito viene mostrato il file di configurazione **EthernetKRL** per la comunicazione con il Flexibowl, nominato **ServerKrl.xml**

ServerKrl.xml

```
<ETHERNETKRL>
        <CONFIGURATION>
                <EXTERNAL>
                        <TYPE>Client</TYPE>
                </EXTERNAL>
                <INTERNAL>
                        <IP> 192.168.1.30</IP>
                        <PORT>54600</PORT>
                        <ALIVE Set_Flag="1" />
                </INTERNAL>
        </CONFIGURATION>
        <RECEIVE>
                <RAW>
                        <ELEMENT Tag="Buffer" Type="STREAM" Set_Flag="2" EOS="10,13"/>
                </RAW>
        </RECEIVE>
            <SEND />
</ETHERNETKRL>
```

Configurazione del Plug-In

Step 4.

Di seguito viene mostrato il codice per la comunicazione con Flexivision tramite **EthernetKRL**. Il programma **Flexivision3.SUB**, mostrato di seguito, dovrà essere inserito come task parallelo e l'esecuzione viene regolata tramite un semaforo (**$FLAG[5]**). Il comando da eseguire (in questo codice **"start_Locator"**) deve essere inserito manualmente in **CHAR Request[]** mentre la stringa restituita da Flexivision sarà memorizzata in **Response_[]**. Qualora la richiesta corrisponda ad una posizione **2D** del pezzo (**X,Y,A**), tale posizione verrà memorizzata nelle rispettive componenti (**X,Y,A**) della variabile **6POS FlexivisionPos_**, le rimenenti componenti (**Z,B,C**) dovranno essere inserite (all'interno di tale codice) in **FlexivisionPos_** manualmente in base alle proprie esigenze. Entrambi le variabili **Response_[]** e **FlexivisionPos_** sono variabili globali dichiarate in **Flexivision3.dat**.

Flexivision3.sub

```
&ACCESS RVO
&REL 73
&PARAM DISKPATH = KRC:\R1\System
DEF Flexivision3 ( )
    BOOL cl
    CHAR Bytes[128]
    CHAR Request[128]
    INT initstr
    INT endstr
    INT len_pos
    INT i
    BOOL index
    INT tmp_int
    INT Set_Recipe
    INT Get_Recipe
    INT State_Locator
    INT Start_Empty
    INT Stop_Locator
    INT Start_Locator
    INT Turn_Locator
    INT Test_Locator
    INT Start_Control
    CHAR name[32]
    CHAR xpos[32]
    CHAR ypos[32]
    CHAR rzpos[32]
    CHAR xpos_[32]
    CHAR ypos_[32]
    CHAR rzpos_[32]

    ;VARPOS
    INT OFFSET
    DECL STATE_T STAT
    REAL VAR_X,VAR_Y,VAR_RZ
```

Configurazione del Plug-In

```
;INITIALIZE
FOR i=1 TO (128)
    Bytes[i]=0
    Request[i]=0
    Response_[i]=0
ENDFOR

LOOP

;INIT AND OPEN THE CHANNEL
RET=EKI_Init("ServerKrl")
RET=EKI_Open("ServerKrl")
;WAIT FOR CLIENT CONNECTION
WAIT FOR ($FLAG[1]==TRUE)

;SELECT THE REQUEST
Request[]="start_Locator"

;CONNECTION TRUE
 WHILE ($FLAG[1]==True)
    ;WAIT FOR THE SEMAPHORE
    WAIT FOR (($FLAG[5]==TRUE) or ($FLAG[1]==FALSE))
    IF(($FLAG[5]==TRUE) and ($FLAG[1]==TRUE)) THEN
        ;ANALYZE THE OPERATION TO BE PERFORMED
         Set_Recipe = StrFind(1, Request[], "set_recipe", #NOT_CASE_SENS)
         Get_Recipe = StrFind(1, Request[], "get_recipe", #NOT_CASE_SENS)
         State_Locator = StrFind(1, Request[], "state_locator", #NOT_CASE_SENS)
         Start_Empty = StrFind(1, Request[], "start_empty", #NOT_CASE_SENS)
         Stop_Locator = StrFind(1, Request[], "stop_locator", #NOT_CASE_SENS)
         Start_Locator = StrFind(1, Request[], "start_locator", #NOT_CASE_SENS)
         Turn_Locator = StrFind(1, Request[], "turn_locator", #NOT_CASE_SENS)
         Test_Locator = StrFind(1, Request[], "test_locator", #NOT_CASE_SENS)
         Start_Control = StrFind(1, Request[], "start_control", #NOT_CASE_SENS)

        ;INFO
         ;-----------------------------------------------------------------------------------------------------
        ;SET_RECIPE
        IF(Set_Recipe>0) THEN
            IF($FLAG[1]==TRUE) THEN
                RET = EKI_Send("ServerKrl",Request[])
            ENDIF
            Response_[]="True"
                $FLAG[5]= FALSE
        ENDIF

        ;GET_RECIPE
        IF(Get_Recipe>0) THEN
            IF($FLAG[1]==TRUE) THEN
                RET = EKI_Send("ServerKrl",Request[])
            ENDIF
            WAIT for (($FLAG[2]==TRUE) or ($FLAG[1]==FALSE) )
            IF (($FLAG[2]==TRUE) and ($FLAG[1]==TRUE)) THEN
                RET=EKI_GetString("ServerKrl","Buffer",Bytes[])
                $FLAG[2]=FALSE
                Response_[]=Bytes[]
                $FLAG[5]= FALSE
            ENDIF
        ENDIF
```

Configurazione del Plug-In

```
;STATE_LOCATOR
IF(State_Locator>0) THEN
    IF($FLAG[1]==TRUE) THEN
        RET = EKI_Send("ServerKrl",Request[])
    ENDIF
    WAIT FOR (($FLAG[2]==TRUE) or ($FLAG[1]==FALSE))
    IF (($FLAG[2]==TRUE) and ($FLAG[1]==TRUE)) THEN
        RET=EKI_GetString("ServerKrl","Buffer", Bytes[])
        $FLAG[2]=FALSE
        Response_[]=Bytes[]
        $FLAG[5]= FALSE
    ENDIF
ENDIF

;START_EMPTY
IF(Start_Empty>0) THEN
    IF($FLAG[1]==TRUE) THEN
        RET = EKI_Send("ServerKrl", Request[])
    ENDIF
    WAIT FOR (($FLAG[2]==TRUE) or ($FLAG[1]==FALSE))
    IF (($FLAG[2]==TRUE) and ($FLAG[1]==TRUE)) THEN
        RET=EKI_GetString("ServerKrl","Buffer", Bytes[])
        $FLAG[2]=FALSE
        Response_[]=Bytes[]
        $FLAG[5]= FALSE
    ENDIF
ENDIF

;STOP_LOCATOR
IF(Stop_Locator>0) THEN
    IF($FLAG[1]==TRUE) THEN
        RET = EKI_Send("ServerKrl",Request[])
    ENDIF
    Response_[]="Ok"
    $FLAG[5]= FALSE
ENDIF

;START_LOCATOR, TURN_LOCATOR, TEST_LOCATOR
IF((Start_Locator>0) OR (Turn_Locator>0) OR (Test_Locator>0)) THEN
    IF($FLAG[1]==TRUE) THEN
        RET = EKI_Send("ServerKrl",Request[])
    ENDIF
    index=TRUE
    WHILE index == TRUE
        WAIT FOT ( ($FLAG[2]==TRUE) OR ($FLAG[1]==FALSE))
        IF(($FLAG[2]==TRUE) and ($FLAG[1]==TRUE)) THEN
            RET=EKI_GetString("ServerKrl","Buffer",Bytes[])
            Response_[]=Bytes[]
            $FLAG[2]=FALSE
            ;IN ERROR
            IF(StrFind(1, Response_[], "#", #NOT_CASE_SENS)>0) THEN
            index = FALSE
            $FLAG[5]= FALSE
            halt
```

Configurazione del Plug-In

```
;HOPPER
ELSE
   IF(StrFind(1, Response_[], "Hopper", #NOT_CASE_SENS)>0) THEN
      ;CONTROL THE HOPPER
       PULSE($OUT[33],TRUE,0.5)
       index = TRUE
    ELSE
       initstr=1
       endstr=1
       endstr = StrFind(initstr, Bytes[], ";", #NOT_CASE_SENS)
       FOR i=initstr TO endstr-1
          tmp_int= StrAdd(name[], Bytes[i])
       ENDFOR

       ;XPOS
       initstr=endstr+1
       endstr = StrFind(initstr, Bytes[], ";", #NOT_CASE_SENS)
       endstr=endstr+(initstr-1)
       FOR i=initstr TO endstr-1
          tmp_int= StrAdd(xpos[], Bytes[i])
       ENDFOR
       ;YPOS
       initstr=endstr+1
       endstr = StrFind(initstr, Bytes[], ";", #NOT_CASE_SENS)
       endstr=endstr+(initstr-1)
       FOR i=initstr TO endstr-1
          tmp_int= StrAdd(ypos[], Bytes[i])
       ENDFOR
       ;RZPOS
       initstr=endstr+1
       endstr = StrFind(initstr, Bytes[], ";", #NOT_CASE_SENS)
       endstr=endstr+(initstr-1)
       FOR i=initstr TO (endstr-1)
          tmp_int= StrAdd(rzpos[], Bytes[i])
       ENDFOR

       ;XPOS_
       initstr=1
       endstr = StrFind(initstr, xpos[], ",", #NOT_CASE_SENS)
       len_pos=StrLen(xpos[])
       IF (endstr>0) THEN
          FOR i=initstr TO endstr-1
             tmp_int=StrAdd(xpos_[], xpos[i])
          ENDFOR
          tmp_int=StrAdd(xpos_[],".")
          FOR i=endstr+1 TO len_pos
             tmp_int=StrAdd(xpos_[], xpos[i])
          ENDFOR
       ELSE
          FOR i=initstr TO len_pos
             tmp_int=StrAdd(xpos_[], xpos[i])
          ENDFOR
       ENDIF
```

Configurazione del Plug-In

```
;YPOS_
initstr=1
endstr = StrFind(initstr, ypos[], ",", #NOT_CASE_SENS)
len_pos=StrLen(ypos[])
IF (endstr>0) THEN
    FOR i=initstr TO endstr-1
        tmp_int=StrAdd(ypos_[], ypos[i])
  ENDFOR
   tmp_int=StrAdd(ypos_[],".")
  FOR i=endstr+1 TO len_pos
      tmp_int=StrAdd(ypos_[], ypos[i])
  ENDFOR
ELSE
    FOR i=initstr TO len_pos
        tmp_int=StrAdd(ypos_[], ypos[i])
    ENDFOR
ENDIF
;RZPOS_
initstr=1
endstr = StrFind(initstr, rzpos[], ",", #NOT_CASE_SENS)
len_pos=StrLen(rzpos[])
IF (endstr>0) THEN
    FOR i=initstr TO endstr-1
        tmp_int=StrAdd(rzpos_[], rzpos[i])
    ENDFOR
    tmp_int=StrAdd(rzpos_[],".")
    FOR i=endstr+1 TO len_pos
        tmp_int=StrAdd(rzpos_[], rzpos[i])
    ENDFOR
ELSE
    FOR i=initstr TO len_pos
        tmp_int=StrAdd(rzpos_[], rzpos[i])
    ENDFOR
ENDIF

;FROM STRING TO REAL
OFFSET = 0
SREAD (xpos_[], STAT, OFFSET, "%10f",VAR_X)
OFFSET = 0
SREAD (ypos_[], STAT, OFFSET, "%10f",VAR_Y)
OFFSET = 0
SREAD (rzpos_[], STAT, OFFSET, "%10f",VAR_RZ)

FlexivisionPos_.x=VAR_X
FlexivisionPos_.y=VAR_Y
FlexivisionPos_.z=8.5 ;inserire la quota di presa
FlexivisionPos_.A=VAR_RZ
FlexivisionPos_.B=0
FlexivisionPos_.C=180

index = FALSE
```

Configurazione del Plug-In

```
                        ENDIF
                    ENDIF
                ELSE
                    index=FALSE
                ENDIF
            ENDWHILE
        ENDIF
    ENDIF
ENDWHILE
RET = EKI_ClearBuffer("ServerKrl",Bytes[])
RET=EKI_Clear("ServerKrl")
WAIT FOR $FLAG[1]==False
ENDLOOP
END
```

Step 5.

Flexivision3.dat

```
&ACCESS RVP
&PARAM DISKPATH = KRC:\R1\System
&REL 73
DEFDAT FLEXIVISION3 PUBLIC
DECL EKI_STATUS RET
DECL GLOBAL POS FlexivisionPos_={X -103.620,Y 65.0400,Z 8.50000,A 0.0,B 0.0,C
180.000}
DECL CHAR Response_[128]
ENDDAT
```

Configurazione del Plug-In

Step 6.

Di sequito viene riportato un esempio di utilizzo del programma **Flexivision3.sub** tramite un semplice programma di pick and place con l'utilizzo della ventosa come gripper, quest'ultimo controllato attraverso **$OUT[1]** . Dopo il deposito del pezzo viene attivato un soffio d'aria per il posizionamento del pezzo controllato tramite **$OUT[2].**

Pick_place.src

```
&ACCESS RVP
&REL 223
&PARAM DISKPATH = KRC:\R1\Program
DEF pick_place( )
      EXT BAS (BAS_COMMAND :IN,REAL :IN )
      DECL POS P1
      ;APPROACH/DEPART POINT
      DECL POS Pos_Pick_trasl
      BAS(#INITMOV,0)
      $BASE=Base_data[1]
      $TOOL=tool_data[2]
      $APO.CPTP=50

      ;WAREHOUSE POSITION
      P1= {X -413.31, Y 271.25, Z -41.74, A 0.00, B -90.00, C -35.42, S 0, T 10}
      PTP P1
      ;FLAG[5] "ACTIVE" Flexivisione3.sub
      $FLAG[5]=TRUE

      LOOP
         $OUT[1]=FALSE
         WAIT for $FLAG[5]==FALSE
         TRIGGER WHEN DISTANCE=0 DELAY=100 DO
         $OUT[2]=TRUE  TRIGGER WHEN DISTANCE=1 DELAY=0 DO
         $out[2]=FALSE
         ; UPDATE THE APPROACH/DEPART POINT
         Pos_Pick_trasl= FlexivisionPos_
         Pos_Pick_trasl.z= -40 ;update the z component for the approach point

         PTP Pos_Pick_trasl C_PTP
         TRIGGER WHEN DISTANCE=0 DELAY=130 DO $OUT[1]=TRUE
         PTP FlexivisionPos_
         PTP Pos_Pick_trasl C_PTP
         TRIGGER WHEN DISTANCE =1 DELAY=-130 DO
         $FLAG[5]=TRUE  PTP P1
    ENDLOOP
END
```

Lista Comandi FlexiVision

Per inviare il comando a FlexiVision è necessario modificare il valore della stringa "command".

N_Mission	Command	Action
1	“start_Locator"	Starts the parts localization process by recalling the FlexiBowl handling routine in case there are no parts that can be picked up. ***Return:*** “Pattern1;x;y;r”.
2	“stop_Locator"	Stops the process of locating the object with the aid of the FlexiBowl.
3	“turn_Locator"	If no parts are picked up, by this command the operator can make the Flexibowl rotate and the “start_Locator” routine start. ***Return:*** “Pattern1;x;y;r”.
4	“test_Locator"	Starts the process of locating the object without the aid of the FlexiBowl. ***Return:*** “Pattern1;x;y;r”.
5	“start_Control"	Starts the inspection cycle. ***Return:*** “Control1;x;y;r”.
6	“state_Locator"	Locator status diagnostics is shown: ***Return:*** “Locator is Running” “Locator is in Error” “Locator is not Running”.
7	“start_Empty"	Start the FlexiBowl® Quick-Emptying sequence. ***Return:*** “start_Empty ended”
8	“get_Recipe"	The name of the recipe currently loaded on FlexiVision is shown. ***Return:*** “recipe name”.
9	“set_Recipe=recipe name"	The recipe corresponding to the sent "recipe name" is loaded.