Manual

FlexiVision

KUKA PLUG-IN

ars | Feeding Industrial Robotics

INDEX

This Plugin was developed with the idea of communicating **quickly and safely** with the Flexivision through **Kuka** robots. The Plugin requires the **KUKA Ethernet KRL licence** to work correctly.

FlexiBowl® Plug-In

Plug-In Configuration

Step 1.

KUKA.Ethernet KRL overview

KUKA.Ethernet KRL **functions** is a rechargeable technology package with the following functions:

- Data exchange via the **EKI**
- Receiving **XML** data from an external system
- Sending **XML** data to an external system
- Receiving binary data from an external system
- Sending binary data to an external system

Properties

- Robot control and external system as client or server
- Configuring connections using the XML-based configuration file
- Configuring **"Event messages"**
- Checking connections by pinging the external system.
- Reading and writing data of the Submit interpreter
- Reading and writing data of the robot interpreter

Communication The data is transferred via **TCP/IP** protocol. The **UDP/IP** protocol can be used, but it is not recommended (network protocol without connection, e.g. no data loss detection).

Plug-In Configuration

Step 2.

Configuring an Ethernet connection

Overview

An Ethernet connection is configured via an X**M**L file.

A configuration file must be defined for every connection,

in the **C:\KRC\ROBOTER\Config\User\Common\EthernetKRL**

folder of the robot control.

The XML file name is simultaneously the login in **KRL**.

Example: …\EXT.XML —> EKI_INIT("EXT")

XML structure for connection characteristics

Description

The settings for the external system can be defined in the

<EXTERNAL>… </EXTERNAL> section:

The XML files are **"case sensitive"**. Consider upper/lower case.

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

Plug-In Configuration

Step 3.

Below is shown the configuration file for the **EthernetKRL** communication, called **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>
```

Plug-In Configuration

Step 4.

Below is shown the code for the communication with the Flexibowl via **EthernetKRL**.
The program **Flexivision3.SUB** has to be put in a parallel task and its executions managed with a semaphore (**$FLAG[5]**). The command to execute (in this code **"start_Locator"**) has to be write into **CHAR Request[]** while the Flexivision's response is set into **Response_[]**. If the request is a **2D** pattern's position (**X,Y,A**), this position is set into (**X,Y,A**) components of **6POS FlexivisionPos_**, the other (**Z,B,C**) components have to be manually update into **FlexivisionPos_** (in this code) . Both **Response_[]** and **FlexivisionPos_** are global variables and they are declared into **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
```

Plug-In Configuration

```
;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
```

Plug-In Configuration

```
;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
```

Plug-In Configuration

```
;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
```

Plug-In Configuration

```
;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
```

Plug-In Configuration

```
                    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
```

Plug-In Configuration

Step 6.

Below is showed an example to use the program **Flexivision3.sub** throught a simple pick and place program with a sucker gripper. The gripper is managed throught the **$OUT[1]**. After the place of pattern an air blow is activate throught **$OUT[2]** for the positioning of pattern.

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
```

FlexiVision Command List

To send the command to FlexiVision you must modify the value of the "command" string.

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.