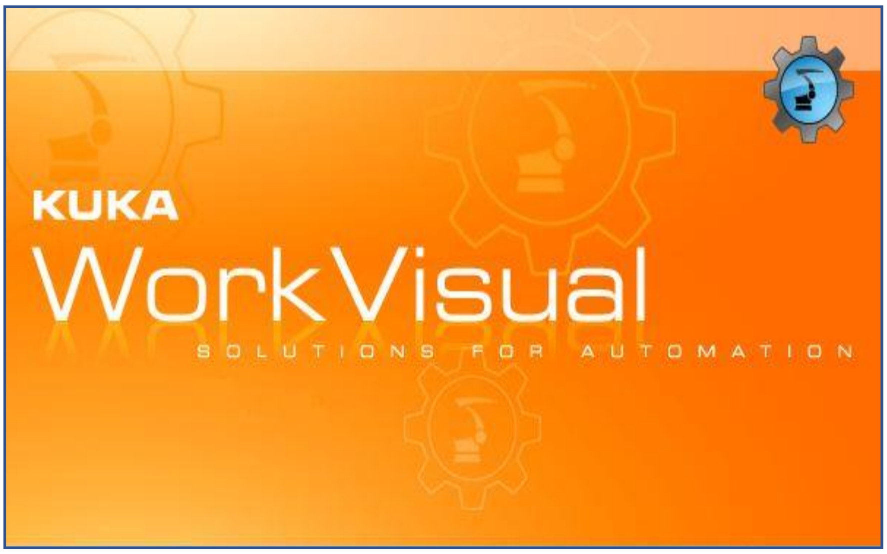
KUKA FLEXIBOWL PLUGIN

This Plugin was developed with the idea of communicating quickly and safely with the flexibowl through Kuka robots, using Work Visual 5 software.
The Plugin requires the KUKA Ethernet KRL licence to function correctly.



Work
Visual

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).

STEP 2:

Configuring an Ethernet connection

Overview An Ethernet connection is configured via an XML 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:

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>
```

Work
Visual

STEP 3:

Below we will show you how to create the EthernetKRL configuration file, called ServerKrl.xml

```
ETHERNETKRL>
   <CONFIGURATION>
      <EXTERNAL>
         <TYPE>Client</TYPE>
      </EXTERNAL>
      <INTERNAL>
        <ENVIRONMENT>Submit</ENVIRONMENT>
         <IP>192.168.1.10</IP>
         <PORT>77775</PORT>
         <ALIVE Set_Flag="2"/>
         <Messages Display="disabled" Logging="error"/>
      </INTERNAL>
   </CONFIGURATION>
   <RECEIVE>
      <RAW>
         <ELEMENT Tag="Buffer" Type="STREAM" Set_Flag="1" Size="64" EOS="13,10" />
      </RAW>
   </RECEIVE>
   <SEND>
      <RAW>
         <ELEMENT Tag="Buffer" Type="STREAM" Set_Flag="1" Size="64" EOS="13,10"/>
      </RAW>
   </SEND>
</ETHERNETKRL>
```

STEP 4:

Below is the code for communication with the Flexibowl via the EthernetKRL. Such a script can be launched for the Flexibowl movement or put in a parallel task, and the execution set via a semaphore. The code will receive a command to execute and will return a string with the response from the flexibowl.

```
&ACCESS  RVO
&COMMENT USER specified PLC program
DEF  FlbPlugin ( )
   DECL EKI_STATUS RET

   CHAR Bytes[100]
   CHAR TMP[100]
   CHAR command[100]
   int lenght
   char returnFlb[100]
   int returnok
   int movment
   int found
   CHAR moving [24]

LOOP

 ;INITIALISE
 FOR i=(1) TO (128)
   Bytes[i]=0
  ENDFOR

 ;flags to start movement
 wait for $FLAG[3]

 ;set the command to be sent
 ;for example
 command="QX2"

 ;CREATE THE STRING (CHAR(0)+CHAR(7)+COMMAND+CHAR(13))
 lenght = StrLen(command[])
 TMP[]="0"
 returnok = stradd(Bytes[],TMP[])
 TMP[]="7"
 returnok = stradd(Bytes[],TMP[])
 TMP[]=command[]
 returnok = stradd(Bytes[],TMP[])
 TMP[]="13"

 returnok = stradd(Bytes[],TMP[])

 ;enable the connection
 RET=EKI_Init("ServerKrl")
 RET=EKI_Open("ServerKrl")
 ;while waiting for communication from a client
 WAIT FOR $FLAG[2]
 ;send the command to the flexibowl
 RET = EKI_Send("ServerKrl",Bytes[])
;wait for the response from the flexibowl
 WAIT FOR $FLAG[1]
 RET=EKI_GetString("ServerKrl","Buffer",Bytes[])

 ;analyse the command sent
 movment = StrFind(1, command, "QX", #NOT_CASE_SENS)
 returnok = StrFind(1, command, "%", #NOT_CASE_SENS)

 if((movment>0)and(returnok>0)) then
    ;a move command was sent and % replied therefore WAITMOVE
    moving=1;
    While (moving=="1")
       ;INIZIALIZZO
       FOR i=(1) TO (128)
          Bytes[i]=0
       ENDFOR

       ;CREATE THE STRING (CHAR(0)+CHAR(7)+COMMAND+CHAR(13))
       lenght = StrLen("RS")
       TMP[]="0"
       returnok = stradd(Bytes[],TMP[])
       TMP[]="7"
       returnok = stradd(Bytes[],TMP[])
       TMP[]="SC"
       returnok = stradd(Bytes[],TMP[])
       TMP[]="13"
       returnok = stradd(Bytes[],TMP[])
       ;send the command to the flexibowl
       RET = EKI_Send("ServerKrl",Bytes[])

      ;wait for the response from the flexibowl
      WAIT FOR $FLAG[1]

      RET=EKI_GetString("ServerKrl","Buffer",Bytes[])
      found = StrFind(1, Bytes[], "F")
      if(found>0) then
         moving=1
      else
         moving=0
      endif

    endwhile
    returnFlb="Done"
else
    returnFlb=Bytes[]
endif
    RET = EKI_ClearBuffer("ServerKrl",Bytes[])

ENDLOOP

END
```

Work
Visual

STEP 5:

Lista dei comandi:

Action	Description
MOVE	Moves the feeder the current parameters.
MOVE-FLIP	Moves the feeder and activates Flip simultaneously
MOVE-BLOW-FLIP	Moves the feeder and activates Flip and blow simultaneously
MOVE-BLOW	Moves the feeder and activates Flip simultaneously
SHAKE	Shakes the feeder with the current parameters
LIGHT ON	Light on
LIGHT OFF	Light off
FLIP	Flip
BLOW	Blow
QUICK_EMPTING	Quick Emptying Option
RESET_ALARM	Reset Alarm and enable the motor

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