04/05/2021

Praktischer Teil der Kommunikationsphase mit einem einfachen Beispiel

Yahya Fakhet :7012464

Faissal Hammouda :7012301

WS/2021

Einführung:

Dieses Dokument dient zur Erläuterung der Kommunikationsschritte zwischen dem Beckhoff XTS (Extended Transport System) und dem KUKA Roboter KRC4 Kompakt.

Das folgende Diagramm zeigt den Informationsablauf zwischen den letzten beiden Maschinen und einem Rechner unter Beachtung des OPC UA-Kommunikationsprotokolls.

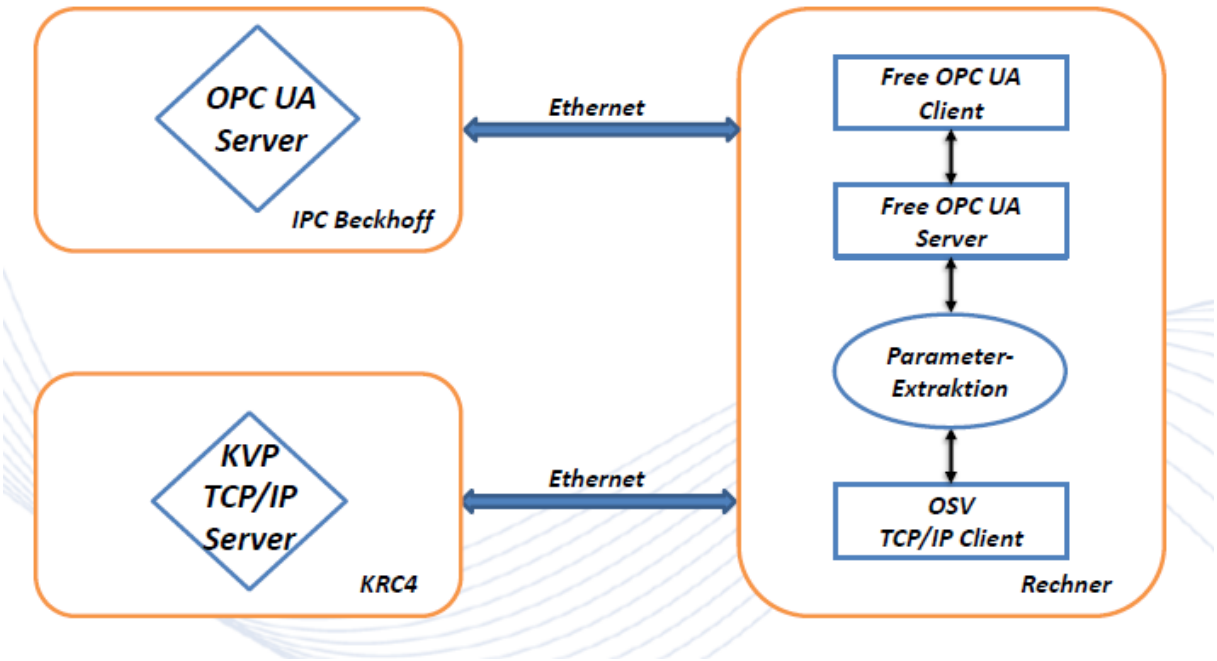


Abbildung 1: Informationsablauf

1. ***KUKAVARPROXY (KVP):***

KUKAVARPROXY ist ein TCP/IP-Server, der auf Netzwerknachrichten am TCP-Port 7000 hört

und Daten in die KRC-Systemvariablen liest und schreibt.

Um eine Ethernet-Verbindung (TCP/IP) herzustellen, muss zuerst KUKAVARPROXY auf dem Betriebssystem der Steuerung ausgeführt und dann die Netzwerkverbindung vom KUKA ‚HMI' konfiguriert werden.

Die praktischen Schritte sind wie folgt:

- ➢ Kopieren KUKAVARPROXY in das Betriebssystem auf der KRC

- ✓ Erhalten den KUKAVARPROXY auf: https://sourceforge.net/projects/openshowvar/files/openshowvar/REV%200.13.0/kukavarproxy-6_1.rar/download
- ✓ Entpacken und kopieren den Ordner auf einen USB-Stick

- ✓ USB-Stick an die KRC anschließen
- ✓ Als Experte oder Administrator anmelden: KUKA Menü -> Konfiguration -> Benutzergruppe
- ✓ HMI minimieren: KUKA Menü -> Inbetriebnahme -> Service -> HMI minimieren
- ✓ Kopieren den Ordner KUKAVARPROXY auf den Desktop
- ✓ KUKAVARPROXY.exe starten
- ✓ Wenn es ein Problem mit der Datei cswsk32.ocx gibt, verwenden den folgenden Befehl in der Eingabeaufforderung des Administrators regsvr32.exe c:\asdf\cswsk32.ocx und ändern das asdf in den tatsächlichen Dateipfad.
- ✓ Starten das Programm automatisch beim Neustart, indem eine Verknüpfung zu KUKAVARPROXY.exe in Windows Start -> Alle Programme -> Rechtsklick auf Autostart -> Öffnen definieren

- ➢ HMI-Netzwerk-Konfiguration
  - ✓ Verbinden den Roboter mit einem Netzwerk
  - ✓ Konfigurieren die IP-Adresse der KRC: KUKA Menü -> Inbetriebnahme -> Netzwerkkonfiguration
  - ✓ Entsperren Sie Port 7000: KUKA Menü -> Inbetriebnahme -> Netzwerkkonfiguration -> Erweitert
  - ✓ NAT -> Port hinzufügen -> Portnummer 7000
  - ✓ Einstellen der erlaubten Protokolle: TCP/UDP

- ➢ Kommunikations-Anschluss: KUKA Line Interface X66
  - ✓ Der X66-Anschluss auf dem Anschlussfeld ist für den Anschluss eines externen Computers für Installations-, Programmier-, Debugging- und Diagnosezwecke vorgesehen

2. OpenSowVar (OSV):

OSV läuft als Client auf einem entfernten Rechner, der über TCP / IP mit der KUKA KRC4-Steuerung verbunden ist.

KVP ist der TCP/IP-Server, der auf der KRC4 des Roboters läuft, während OSV der Client ist, der sich mit dem Server verbindet.

der Link von OpenShowVar auf der Github-Seite ist der folgende:

https://github.com/linuxsand/py_openshowvar

Um das Python OSV auf dem Rechner zu installieren, muss zuerst die Python-Bibliothek ‚*py_openschowvar*' installiert werden (**pip install py_openshowar**)

Nach der Installation von *py_openshowvar* wird mit Hilfe von Python-Befehlen ein Test der Kommunikation zwischen dem Server und dem Client durchgeführt.

Die Kodierung des Kommunikationstests zwischen dem KVP-Server und dem OSV-Client ist wie folgt:

>>> from py_openshowvar import openshowvar

>>> client = openshowvar('172.31.1.147', 7000)

>>> client.can_connect

True

‚True' zeigt an, dass die Verbindung bereits aufgebaut ist.

3. Free OPC UA Python (Server/Client):

der Link von Free OPC UA Python (Server/Client) auf der Github-Seite ist der folgende:

https://github.com/FreeOpcUa/python-opcua

4. Kommunikations-Test:

4.1. Zusammenführen des Kommunikationscodes zwischen OpenShowVar und dem OPCUA Python Server in eine einzige Python-Datei:

```
from opcua import Server #Server importieren aus der OPCUA-Python-Bibliothek
from kukavarproxy import * # importieren der KUKAVARPROXY-Bibliothek

server = Server()      # Erstellen eines OPCUA-Instanzservers

server.set_endpoint("opc.tcp://127.0.0.1:8080") # Server-Adresse des Endpunkts
einstellen

addspace = server.register_namespace( "OPC_SIMULATION_SERVER") # Namensbereich des
Servers registrieren

node = server.get_objects_node() # Objektknoten vom Server holen

Param = node.add_object(addspace, "Parameters") # Werteinstellungen für
Serverknoten erstellen

M1 = Param.add_variable(addspace,"M1",0) # Initialisierungsvariable für den Knoten
"Server-Einstellungen" hinzufügen

M1.set_writable() # Hinzufügen der Berechtigung schreibbar für die Variable

server.start() # Starten des Servers


k = KUKA('172.31.1.147') # KUKA Objekt erstellen und verbinden (mit Adresse IP
Verbindung)

# Test-Beispiel

i = 0 # Initialisierung der Position für M1 im Server
```

```
while True:
    if i != M1.get_value():

        i=M1.get_value() # Bereitschaft für Client M1 Aktion

        k.write('M',i) # Senden der M-Aktion an Kuka (Ändern des M-Werts)
        # Serverwerte zurücksetzen, um eine Ersetzung vorzunehmen
        i = 0
        M1.set_value(0)
```

4.2. OPCUA Python Client Kommunikationscode

```
# Client UA importieren aus der OPCUA-Python-Bibliothek
import sys
sys.path.insert(0, "..")
from opcua import Client,ua

# OPCUA Python-Server-Adresse aufrufen
url = "opc.tcp://127.0.0.1:8080"

# OPCUA TwinCat 3 Server-Adresse aufrufen
url2 = "opc.tcp://172.31.1.149:4840"

# Kuka-Client-Instanziierung
client_kuka = Client(url)

# TwinCat3-Client-Instanziierung
client_xts = Client(url2)

# Kuka-Client-Verbindung
client_kuka.connect()

# TwinCat3-Client-Verbindung
client_xts.connect()

print("Client Connected")

# Wert, der behandelt werden soll
kuka_pos = client_kuka.get_node("ns=2;i=2") # ns=2;i=2 nodeID
M = client_xts.get_node("ns=4;s=ua.M") # ns=4;s=ua.M nodeID ua.M

while True:
        print(M.get_value())
        pos = input('set kuka pos 1/2')
        # KUKA Aktionswert senden
        kuka_pos.set_value(pos)
        # Senden M twinCat3 OPCUA Wert
        M.set_attribute(ua.AttributeIds.Value, ua.DataValue(6))
```

4.3. TwinCat 3 OPCUA Server

➢Bereitstellen des OPCUA TwinCAT-Servers

• Importieren die Daten in den OPCUA-Server

• Aktivieren die TF6100-Lizenz für den Server

➢Konfigurieren die verschiedenen Facetten des OPCUA-Servers

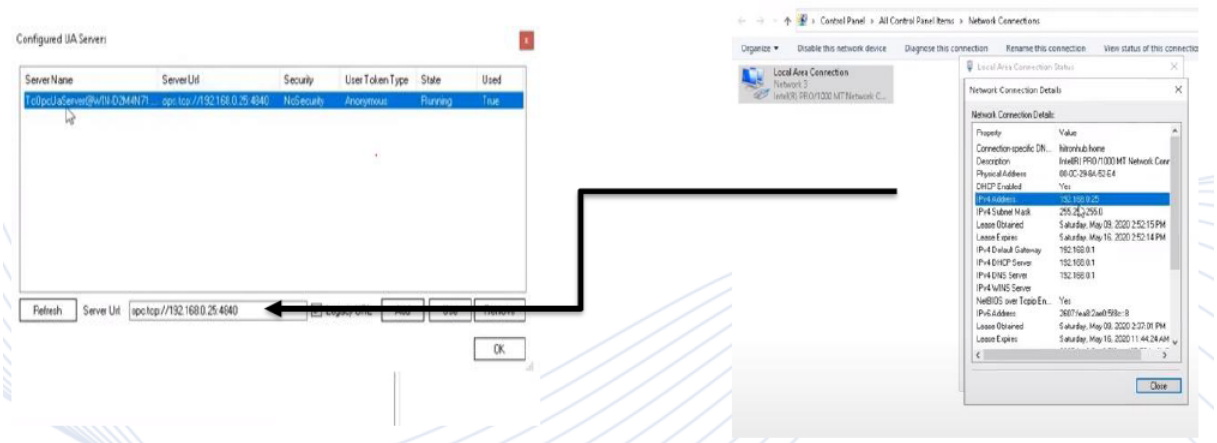
• Nachdem ein Projekt abgeschlossen ist. Es wird eine Verbindung zwischen dem OPCUA-Server und den ADS-Geräten hergestellt.

• Es öffnet sich ein Dialogfenster, in dem die Verbindungsparameter konfiguriert werden können, z. B. AMS Net ID, ADS-Port, ...

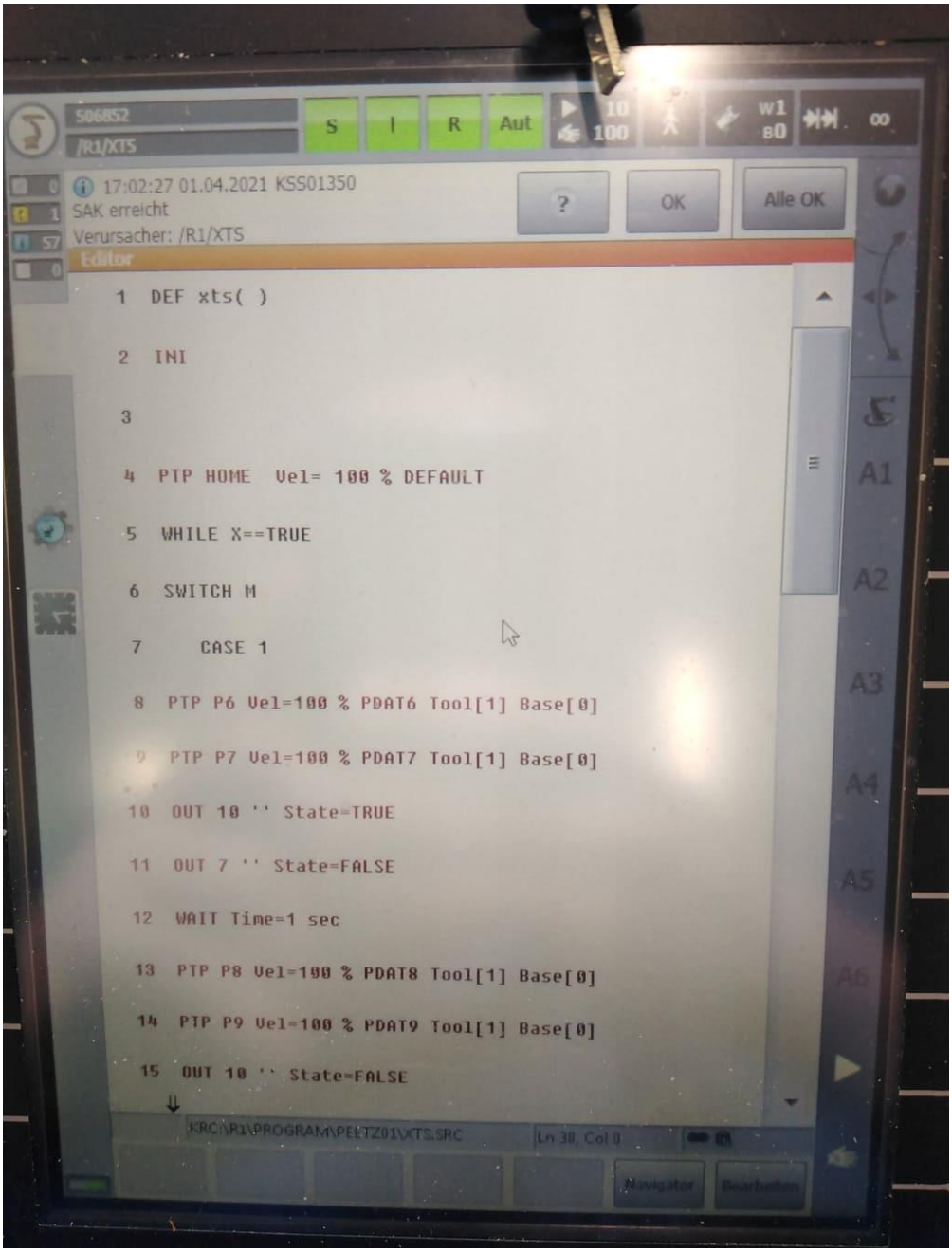
➢Konfigurieren von Endpunkten und Zertifikatsvertrauenseinstellungen

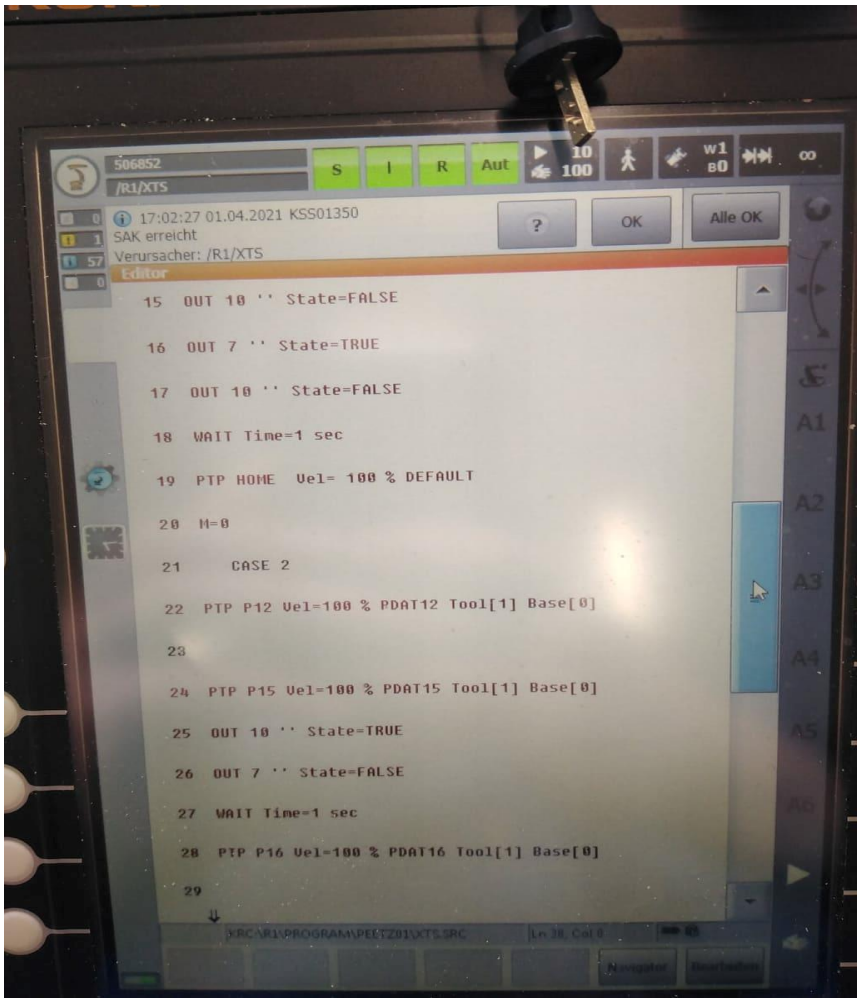
• Die Endpunkte des OPCUA Servers legen fest, welche Sicherheitsmechanismen beim Verbindungsaufbau eines Clients verwendet werden sollen.

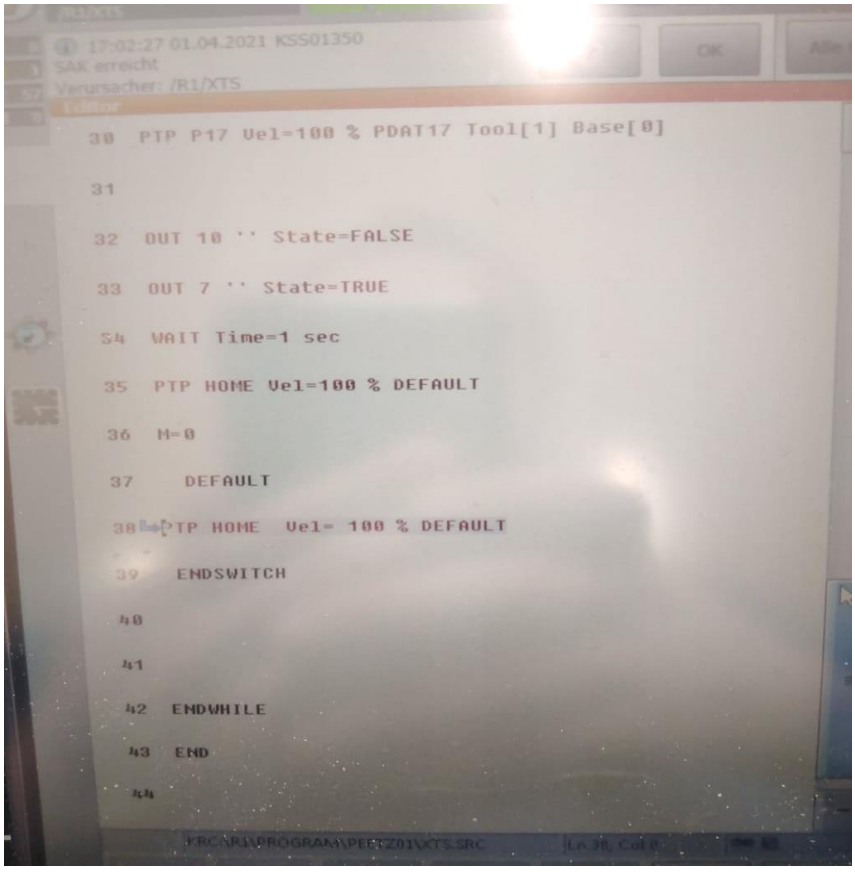
• Aufbau einer Verbindung zum OPCUA-Server: Um eine Verbindung von einem OPCUA Client aufzubauen, muss der Client eine Verbindung mit der URL des OPCUA Servers aufbauen, z.B. opc.tcp://192.168.1.1:4840



4.4. Beispiel für das KRL-Programm des KUKA Roboters




506852
/R1/XTS
S
I
R
Aut
10
100
17:02:27 01.04.2021 KSS01350
SAK erreicht
Verursacher: /R1/XTS
?
OK
Alle OK
Editor
15 OUT 10 '' State=FALSE
16 OUT 7 '' State=TRUE
17 OUT 10 '' State=FALSE
18 WAIT Time=1 sec
19 PTP HOME Vel= 100 % DEFAULT
20 M=0
21 CASE 2
22 PTP P12 Vel=100 % PDAT12 Tool[1] Base[0]
23
24 PTP P15 Vel=100 % PDAT15 Tool[1] Base[0]
25 OUT 10 '' State=TRUE
26 OUT 7 '' State=FALSE
27 WAIT Time=1 sec
28 PTP P16 Vel=100 % PDAT16 Tool[1] Base[0]
29
A1
A2
A3
A4
A5
A6


17:02:27 01.04.2021 KSS01350
SAK erreicht
Verursacher: /R1/XTS
30 PTP P17 Vel=100 % PDAT17 Tool[1] Base[0]
31
32 OUT 10 '' State=FALSE
33 OUT 7 '' State=TRUE
34 WAIT Time=1 sec
35 PTP HOME Vel=100 % DEFAULT
36 M=0
37 DEFAULT
38 PTP HOME Vel= 100 % DEFAULT
39 ENDSWITCH
40
41
42 ENDWHILE
43 END
44