Connection of the operator controls of a handheld terminal to a PLC via Modbus TCP

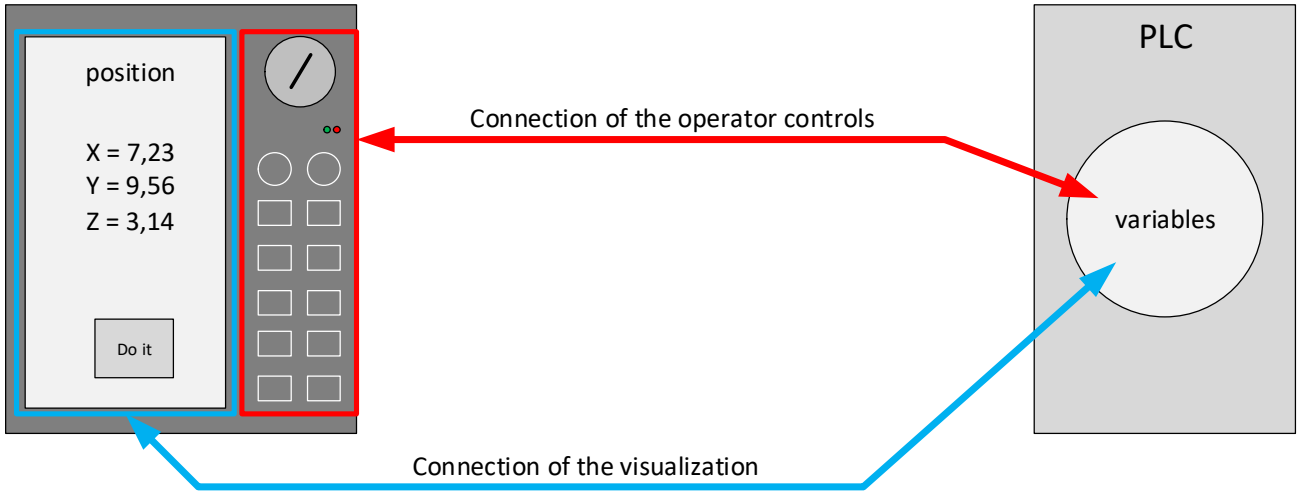
(sg, November 2020)

Test with MachineSoftware V 1.4.2 (aritfactory products/KEBA/Windows-Production)

KEBA
Automation by innovation.

Communication channels between handheld terminal and a PLC

- Connection of the operator controls (= installation elements)
- Connection of the visualization
- Both work on the variables of the PLC
- Connections can use the same or different protocols



KEBA
Automation by innovation.

Training

For connecting the operator controls to a PLC via Modbus TCP

Preconditions

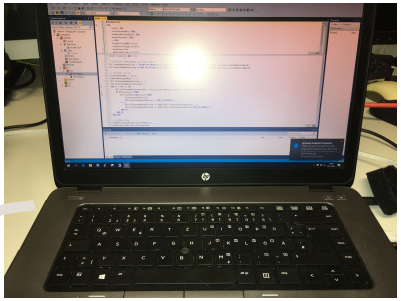
- Basic knowledge of PLCs and Modbus TCP
- Basic knowledge of KEBA handheld terminals

A KemroX PLC with Modbus TCP is used as an example for the training. The handheld terminal serves as Modbus slave, the PLC as Modbus master. The reverse scenario (handheld terminal master, PLC slave) is possible, but is not described further here. However, the findings presented here can also be transferred to other Modbus implementations.

(Building blocks for communication can have different names and interfaces, but the way of thinking should be the same)

Structure

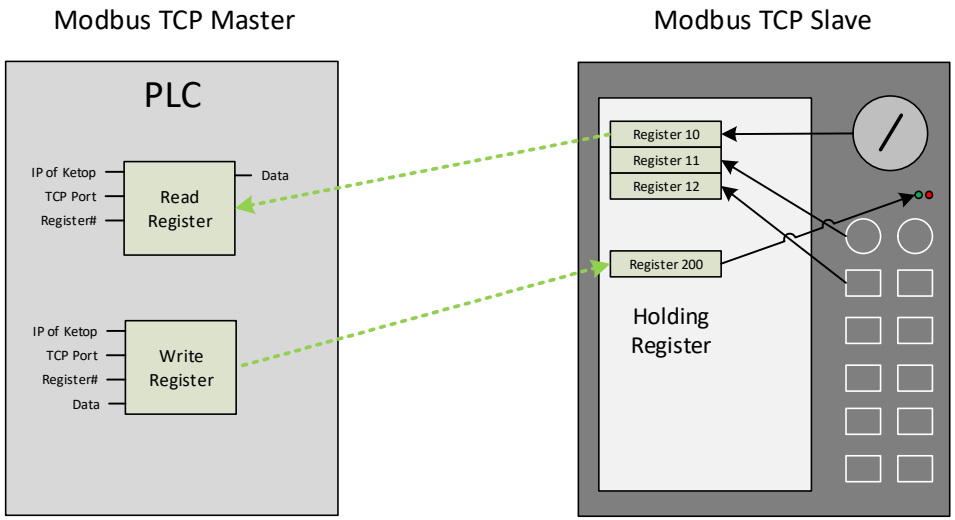
- KEBA PLC with library „*Modbus TCP Client*“
- Handheld terminal with Windows
- Developer PC with KemroX IDE
- All three devices must be connected and available in the network via Ethernet



KEBA
Automation by innovation.

Structure and function of Modbus TCP

- Slave holds values of the operator controls in registers (these are filled by the Modbus interface)
- Slave provides registers for writing (if changed, the value is assigned to the operator control by the Modbus interface)
- Master reads or writes these registers with the help of an IEC block

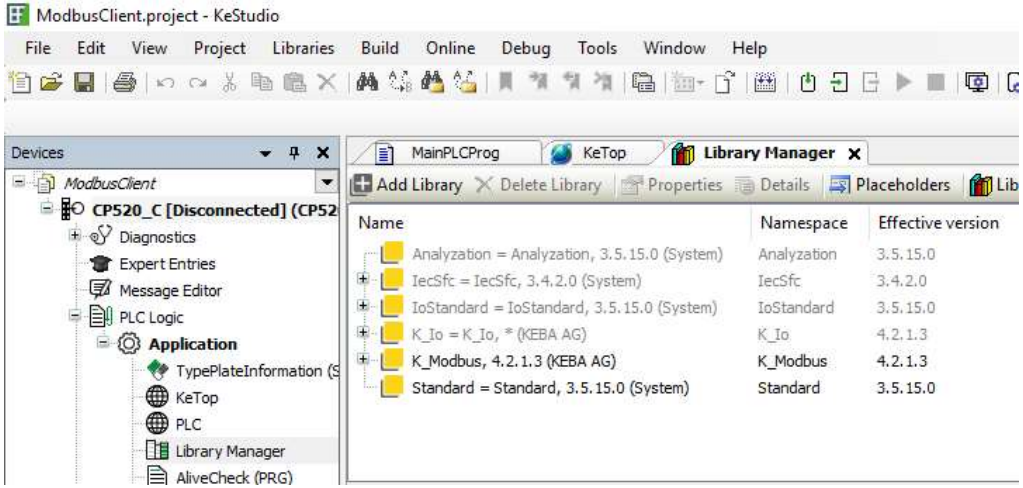


NOTE: For Modbus TCP the Master side must be implemented in the PLC!

PLC project in KemroX

To use Modbus TCP client the Modbus Client Library must be added to the project.

- For this select in the Library Manager with "Add Library" the library K_Modbus in "Application/Technology/Base" and add it to the project.



PLC project in KemroX

Used blocks

```
PROGRAM ModbusClientMain
VAR
    mb_ReadRegs: ModbusReadRegisters;
    mb_WriteRegs: ModbusWriteMultiRegisters;
    mb_Connect: ModbusTCPConnection;
    ....
END_VAR
```

Establishing the connection

```
mb_Connect(Connect:=TRUE,Host:='192.168.214.17',Port:=502);
```

- Slave 192.168.214.17 with Port 502
- Cyclic call of the connection block necessary
- Connection established when mb_Connect.Connected becomes TRUE

Reading the registers

```
mb_ReadRegs(Conn:=mb_Connect,ReadData:=mb_DataRead,Execute:=TRUE,Addr:=0,NbrReg:=123);
```

- Reading the registers 0 to 122 with the previously established connection
- Cyclic call of the reading block necessary
- Reading finished when mb_ReadRegs.Done becomes TRUE

Writing the registers

```
mb_WriteRegs(Conn:=mb_Connect,WriteData:=mb_DataWrite_prev,Execute:=TRUE,Addr:=200,NbrReg:=20);
```

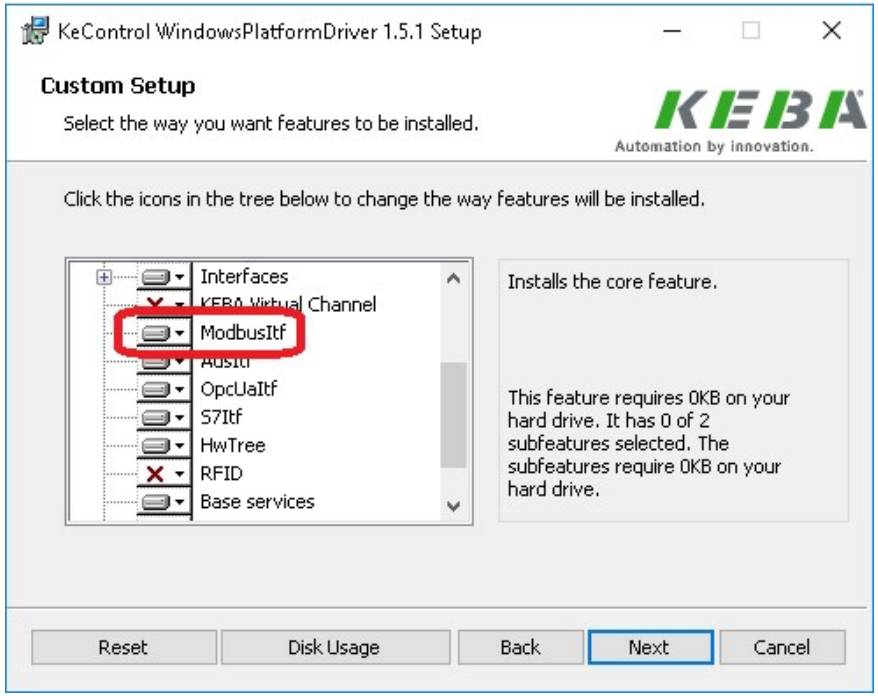
- Writing the registers 200 to 219 with the previously established connection
- Cyclic call of the writing block necessary
- Writing finished when mb_WriteRegs.Done becomes TRUE

Preconditions handheld terminal

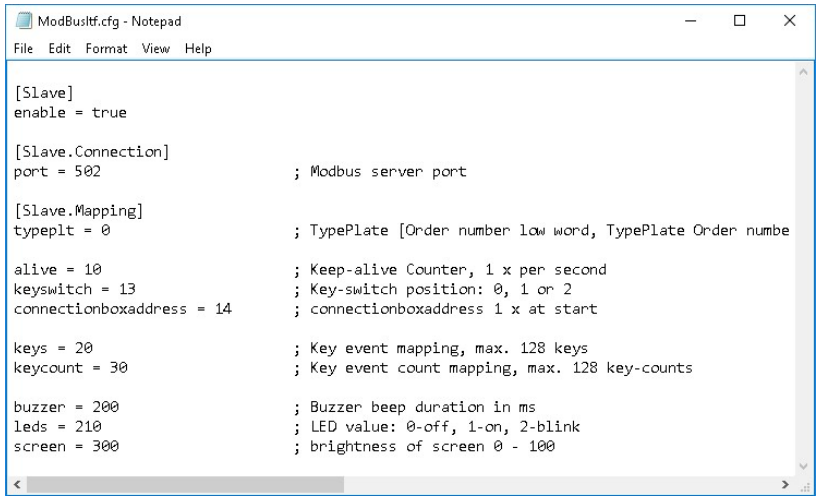
Modbus TCP interface must be installed on the handheld terminal. The installation takes place in the WindowsPlatformDriverSetup.

The WindowsPlattformDriverSetup can be found under „Programs and Features“ gefunden werden.

Only one interface should be used at a time.



Configuration handheld terminal



- Fill ModbusItf.cfg with Notepad
- File is located under
  `C:\ProgramData\KEBA Automation\keview\system`
- Describes communication of the Modbus slave of the handheld terminal
- Describes the connection of operator controls to the registers of the Modbus slave of the handheld terminal

Configuration handheld terminal / communication

```
[Slave]
enable = true

[Slave.Connection]
port = 502                    ; Modbus server port
```

- Enables the communication via Modbus TCP as slave
- Port number via which the Modbus TCP slave can be addressed

Configuration handheld terminal / operator controls

```
[Slave.Mapping]
typeplt = 0

alive = 10
keyswitch = 13
connectionboxaddress = 14
```

- Connection of operator control „alive“ to the register #10

  „alive“ is an operator control that counts up its value every second and is used to check the communication
- Connection of operator control „keyswitch“ to the register #13
- Connection of operator control „connectionboxaddress“ to the register #14
- And so on ...

Configuration handheld terminal / register allocation

(Register numbers are only an example)

```
typeplt = 0

alive = 10
keyswitch = 13
connectionboxaddress = 14

keys = 20
keycount = 30

buzzer = 200
leds = 210
screen = 300
```

typeplt	**0**	order number (low word)
	1	order number (high word)
	2	revision
	3	variant
	4	serial number (low word)
	5	serial number (high word)
- unused -	**6 - 9**	
alive	**10**	
- unused -	**11 - 12**	
keyswitch	**13**	
connectionboxaddress	**14**	
- unused -	**15 - 19**	
keys	**20**	key:0 - key:15
	21	key:16 - key:31
	22	key:32 - key:47
	23	key:48 - key:63
	24	key:64 - key:79
	25	key:80 - key:95
	26	key:96 - key:111
	27	key:112 - key:127
- unused -	**28 - 29**	
keycount	**30**	keycount:0
	31	keycount:1
	32 - 156	keycount:X
	157	keycount:127
- unused -	**158 - 199**	
buzzer	**200**	
- unused -	**201 - 209**	
leds	**210**	led:0
	211	led:1
	212-224	led:X
	225	led:15
- unused -	**226 - 299**	
screen	**300**	screen brightness

- Every register has 16bit (WORD)
- Here is just one example of application
- The allocation of the register is arbitrary
- An operator control with an assignment of more than one register cannot be split (typeplt, keys, keycount, leds)

Configuration handheld terminal / operator controls

Possible configuration entries

```
[Slave.Connection]
  port =
```

```
[Slave.Mapping]
  typeplt:X = "address of variable"
  alive = "address of variable"
  keys:X = "address of variable"
  keycount:X = "address of variable"
  buzzer = "address of variable"
  connectionboxaddress = "address of variable"
  keyswitch = "address of variable"
  enswitch =  "address of variable"
  leds:X =  "address of variable"
  pushbutton = "address of variable"
  pushbuttonled = "address of variable"
  selswitch = "address of variable"
  screen ="address of variable"
  rotswitch = "address of variable"
  pushswitch = "address of variable"
  pushswitchled ="address of variable"
  handwheelposition ="address of variable"
  handwheelpressed ="address of variable"
```

Configuration handheld terminal / operator controls

Possible additional configuration entries not covered in this document (manual handheld terminal as master, PLC as slave)

[Master]

[Master.Connection]

[Master.Mapping]

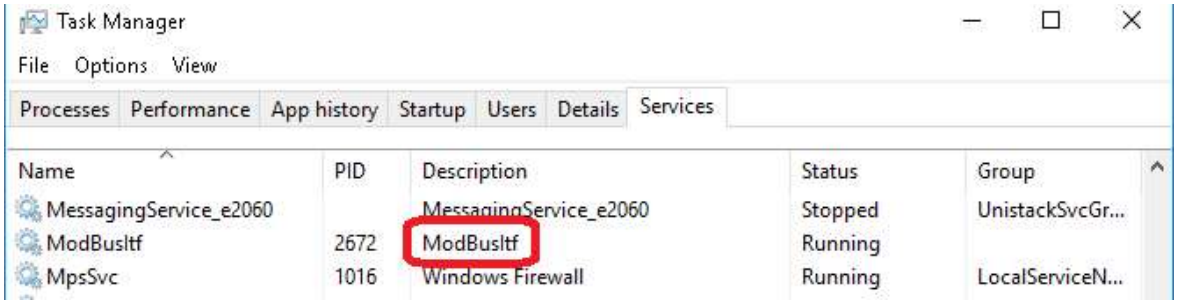
Notes on configuration

- Operator controls not used in the configuration or operator controls that do not exist do not need to be removed from the configuration or commented out. (For reasons of clarity, however, only elements that are actually used should be configured.)

- For a more detailed description, refer to the document "KeTop Modbus protocol". There, all names of the operator controls and their data types are listed.

Start of communication

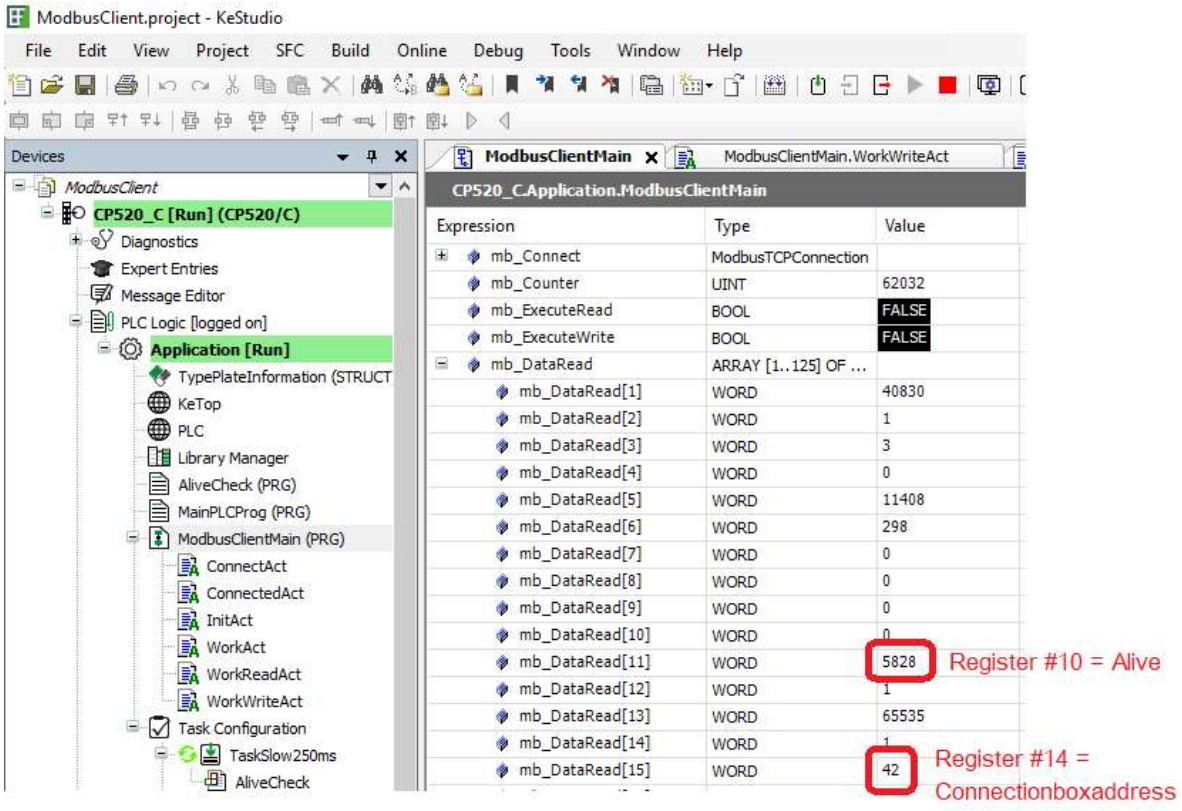
- Download and start the PLC application
- Restart the Modbus TCP communication to accept the new configuration
    - Either restart the handheld terminal
    - Or restart the service „ModBusItf“ in the Taskmanager



- After each change of the configuration, a restart of the communication must be performed, because the configuration is read at startup.

Checking the communication

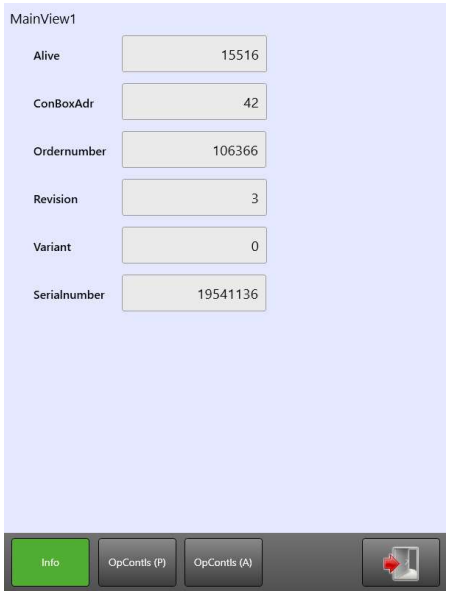
- Start the debugger of the PLC
- Observe the variable into which the registers are written during the read - and there the register for alive (during cyclic reading of the variable the value should change every second)
- A test for the other direction would be to write the register for the buzzer

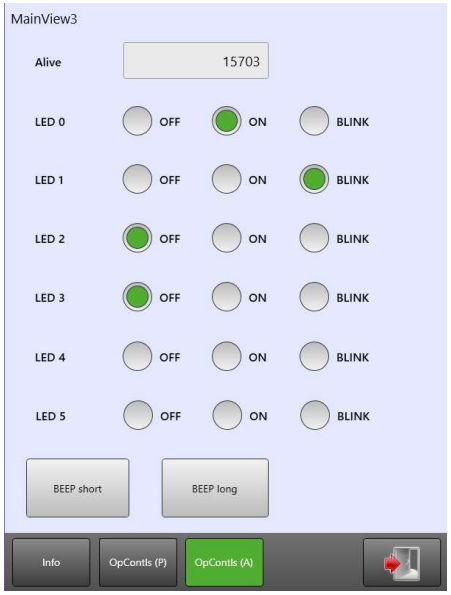


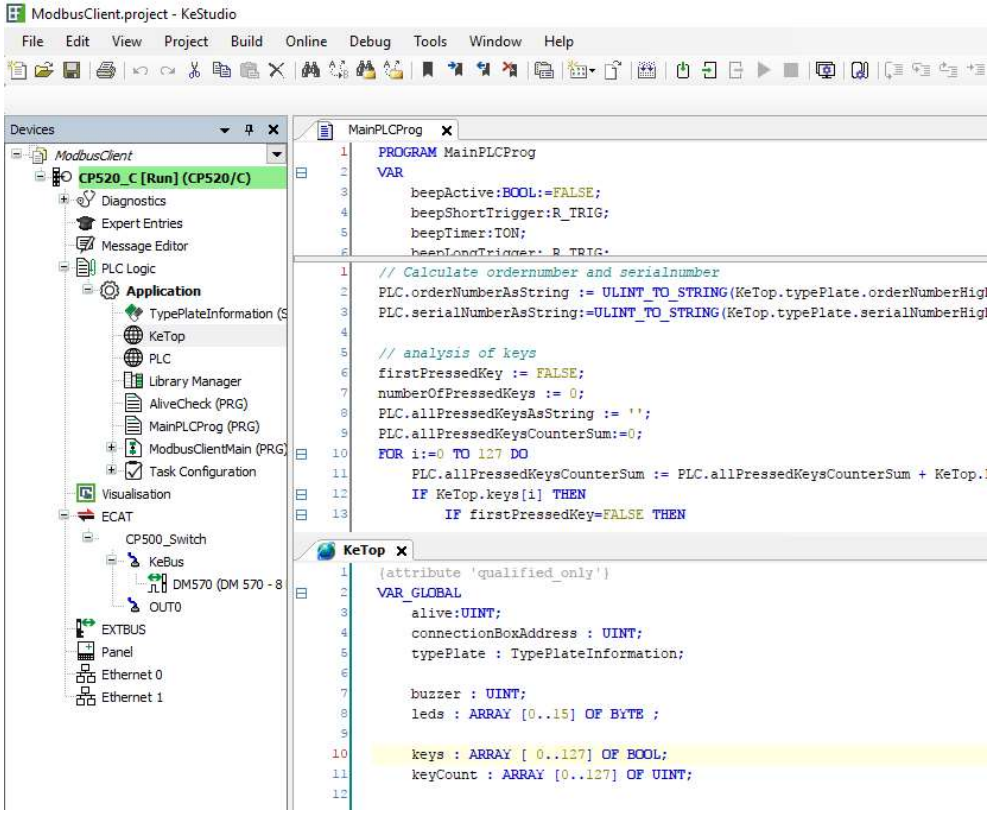
KEBA
Automation by innovation.

Use of the variables

- In the PLC (cyclic copying of the registers into variables of a GVL)
- In the visualization







KEBA
Automation by innovation.